UNIVERSIDAD DE MÁLAGA

E.T.S. INGENIERÍA INFORMÁTICA

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA INFORMÁTICA
GRADUADA/O EN INGENIERÍA DEL SOFTWARE

Aplicación web para novelistas

Escritura en línea y análisis ortográfico

Web application to novelists

Online writing and spell check

Realizado por
Francisco José Rando Márquez

Tutorizado por
Luis Manuel Llopís Torres

Departamento
Lenguajes y ciencias de la computación

UNIVERSIDAD DE MÁLAGA

MÁLAGA, JUNIO DE 2019

Fecha defensa: ___ de julio de 2019

Fdo. El/la Secretario/a del Tribunal

Resumen

Aplicación Web para Novelistas es un servicio software que, como su propio nombre indica, está pensado para apoyar el desarrollo de las novelas, impulsando el proceso creativo para sus escritores. De esta forma, un escritor podrá tener un mayor control para sus obras, tanto a la hora de escribirlas, como a la hora de esquematizarlas y hacerles seguimiento. Para ello se proporcionan una serie de facilidades que van desde una *escritura en línea y análisis ortográfico* con la que crear los capítulos y corregirlos para un libro en concreto, hasta herramientas como la *creación de hojas de personajes, guiones rápidos, notas, planificación de tareas, glosarios o diccionarios personales* con las que el escritor podrá perfilar y centralizar las ideas de su propia obra.

Puesto que la aplicación es el resultado de un proyecto en grupo, este documento en concreto se centra en el desarrollo de la escritura en línea, el almacenamiento de capítulos y libros, y su corrección y análisis de ortografía. A lo largo del mismo, se reflejará cada una de las fases en las que ha consistido este trabajo, la metodología empleada para su desarrollo, así como las tecnologías y recursos utilizados y la experiencia que se ha tenido en su correspondiente uso.

Palabras clave: Análisis ortográfico, escritura en línea, almacenamiento de libros y capítulos, con Angular 5, .Net Core 2.1 y CosmosDB de Azure

Abstract

Web Application for Novelists is a software service that, as its name suggests, is designed to support the development of novels, driving the creative process for their writers. In this way, a writer will be able to have greater control over his or her works, both when writing them and when schematizing and monitoring them. For this purpose, a series of facilities are provided, ranging from online writing and spelling analysis to create chapters and correct them for a specific book, to tools such as the creation of character sheets, quick scripts, notes, task planning, glossaries or personal dictionaries with which the writer can profile and centralize the ideas of his own work.

Since the application is the result of a group project, this particular document focuses on the development of online writing, the storage of chapters and books, and their spelling correction and analysis. Throughout this work, each of the phases in which this work has consisted will be reflected, the methodology used for its development, as well as the technologies and resources used and the experience that has been had in its corresponding use.

Keywords: Spell check, online writing, book storage and chapters, with Angular 5, .Net Core 2.1 and Azure CosmosDB.

Índice

1

Introducción

Motivación

Los escritores manejan una información que va más allá de lo que se ve a simple vista en los capítulos. Escribir un libro es un proceso mental, que al igual que la propia ingeniería del software, pasa por una serie de fases. Recopilación de ideas, objetivos, público dirigido, tipo de narrador, formación de personajes… De hecho, los mapas mentales o conceptuales no son términos que a los escritores les suenen extraño. Hoy en día, los escritores no disponen de ninguna directiva además de su propia experiencia, los hábitos y lo que les inspira y funciona a la hora de crear. Algunos usan libretas para apuntar ideas rápidas, vocablos atractivos, esquemas de sucesos; otros usan el móvil para hacer apuntes rápidos en una nota que en algún futuro puede que usen, arriesgando su vigencia a la propia vida del dispositivo.

Este proyecto surge precisamente para englobar todo este tipo de necesidades y centralizarlas en un único formato, para así impulsar el desarrollo de la obra, y permitir al escritor focalizarse en la escritura; sin que tenga que recurrir a tantas vías externas.

Objetivos

El objetivo fundamental de la aplicación es la facilitación, simplicidad, y proporción de un único formato creativo para el escritor, y esta misma idea es la que se ha conservado para la **escritura en línea**, que tiene como objetivo principal que el escritor no tenga que hacer uso de otras ventanas externas para escribir. Se busca que sea intuitivo, simple y fácil de usar. Sin funcionalidades engorrosas, solo con lo justo y necesario.

Por otro lado, en cuanto a la **gestión de libros**, otro de los objetivos que se quiere alcanzar es que el escritor tenga una imagen atractiva sobre el libro que está escribiendo, y se sienta motivado a seguir dándole forma. Es por eso que cada libro lucirá con su portada, título, descripción y género, haciendo semejanza a cómo sería si estuviese publicado. Se busca que los libros tengan los capítulos bien diferenciados y no sea un documento único en el que escribir, de esta manera el escritor tendrá un mejor acceso y control. Los libros contendrán cada uno de los capítulos creados, pudiendo acceder de manera separada a cada uno, ya que el modelo de escritura está pensado para que el escritor vaya escribiendo capítulo por capítulo.

Esto genera una gran funcionalidad para aquellos escritores más esquemáticos, y es la presentación de un **guion rápido** en el que cada escritor podrá listar los eventos más concretos que deben ocurrir en el capítulo que está desarrollando. De esta forma, se busca conseguir que cada escritor afronte el capítulo cara a cara, con un acceso directo y exclusivo, sin que tenga que estar navegando en el documento del libro entero.

Estructura de la memoria

La estructura de esta memoria seguirá y tratará cada una de las fases de trabajo que se han seguido durante el desarrollo del proyecto:

Especificación de requisitos y casos de uso: En esta fase se concretan los requisitos funcionales que se esperan para la aplicación, las dependencias entre ellos, prioridad y riesgo. Se mostrarán uno a uno durante esta memoria. También, siguiendo el orden de prioridad de los requisitos funcionales, se detallarán los casos de uso propios para la implementación. Se dejará claro lo que se espera de cada requisito y la interacción del usuario con la aplicación.

Planteamiento y preparación de metodología a seguir: Se acordará entre los integrantes la metodología de trabajo con la que desarrollar el proyecto, y preparar su escenario. En la memoria quedará plasmada toda la información de la planificación de trabajo. Cómo ha ido funcionando periodo tras periodo.

Elección y preparación de tecnologías y entornos: Cada una de las tecnologías seleccionadas tendrá un por qué, un funcionamiento y una experiencia de uso que quedarán recogidos en la memoria.

Diseño del modelo de datos: En esta fase quedará reflejado el diseño del modelo de datos, el gestor de base de datos escogido y el motivo de ello, con sus ventajas y desventajas.

Desarrollo de la aplicación: Se explicará cómo se resuelve uno a uno los requisitos funcionales, así como los problemas que se han tenido al intentar resolverlos.

Pruebas: comprobación del correcto funcionamiento de la aplicación.

Posibles extensiones: apartado donde se recopilarán todas las extensiones y mejoras pensadas para la aplicación.

Conclusión: Valores, conocimiento y aportación obtenida en función al resultado y experiencia de desarrollo de la aplicación.

2

Especificación de Requisitos y casos de uso

Para la especificación de requisitos, el equipo se ha basado en la experiencia de varios escritores conocidos, y en particular, la experiencia propia de uno de los integrantes. Estos han sido determinantes para definir los requisitos

funcionales. A continuación, se mostrará una tabla general (tabla 1) de los requisitos del proyecto, de los cuales, solo se explicarán los referentes a las funcionalidades de escritura en línea y gestión de libros.

Requisito	Tratados en este documento	Código
Un usuario podrá registrarse en la aplicación		RF01
El escritor podrá iniciar sesión		RF02
El escritor podrá cerrar sesión		RF03
El escritor tendrá acceso a un perfil propio		RF04
El escritor tendrá acceso a un diccionario personal		RF05
El escritor podrá crear, editar y borrar palabras en su diccionario personal.		RF06
El escritor tendrá acceso a una lista de sus libros	✓	RF07
El escritor tendrá acceso a un glosario asociado a cada libro		RF08
El escritor podrá crear, editar y borrar palabras en sus glosarios		RF09
El escritor tendrá acceso a notas asociadas a cada libro		RF10
El escritor podrá crear, editar y borrar notas en su lista de notas		RF11
El escritor tendrá acceso a hojas de personajes asociadas a sus libros		RF12
El escritor podrá crear, editar y borrar sus hojas de personajes		RF13
El escritor podrá crear, editar y borrar un libro	✓	RF14
El escritor podrá acceder a un libro creado y ver sus capítulos	✓	RF15
El escritor podrá crear y borrar capítulos de un libro	✓	RF16
El escritor tendrá acceso a una pantalla de escritura en línea.	✓	RF17
El escritor podrá editar o escribir capítulos en un editor de texto.	✓	RF18

El escritor podrá elegir opciones de tamaños, estilos y alineación en el editor de texto.	✓	RF19
El escritor podrá exportar su libro a formato Word.	✓	RF20
El escritor podrá hacer una corrección de ortografía de sus capítulos.	✓	RF21
El escritor podrá visualizar en una gráfica el análisis de ortografía de un libro completo.	✓	RF22
El escritor dispondrá de un guion rápido en cada capítulo.	✓	RF23
El escritor podrá acceder a una lista de tareas para planificarse.		RF24
El escritor podrá crear, editar y borrar tareas en su lista de tareas		RF25
El escritor podrá subir y visualizar imágenes de su portada, perfil, y personajes.	✓	RF26
El escritor podrá visualizar en una gráfica el análisis de sentimientos de un libro completo.		RF27

Tabla 1. Requisitos funcionales del proyecto.

Para los requisitos marcados con un '✓', se seguirá la especificación de manera habitual, considerando dependencias entre requisitos, prioridades, y descripción del requisito en cuestión. La información se muestra en las siguientes tablas:

RF07|CU07: El escritor tendrá acceso a una lista de sus libros.

Código	**RF07**
Dependencias	RF04: El escritor tendrá acceso a un perfil propio.
Descripción	El escritor tendrá disponible en su perfil un apartado de libros al que tendrá acceso, donde podrá ver la información principal listada de cada uno de ellos.
Datos específicos	Tendrá que mostrar claramente el título, género, y la portada para cada libro.
Prioridad	*Alta*
Comentarios	-

Tabla requisito: RF07

Código	**CU07**	
Requisitos asociados	RF07: El escritor tendrá acceso a una lista de sus libros.	
Descripción	El escritor tendrá disponible en su perfil un apartado de libros al que tendrá acceso, donde podrá ver la información principal listada de cada uno de ellos.	
Objetivo	Mostrar al escritor la lista de sus libros.	
Precondición	El usuario debe estar registrado.	
Secuencia normal	**Paso**	**Acción**
	1	El escritor selecciona su perfil.
	2	El escritor visualiza los libros que tiene creados.
Postcondición	Se visualizan los libros creados por el escritor en la aplicación, o en caso de que no hay ninguno, un mensaje indicándolo.	
Excepciones	**Paso**	**Excepción**
	1	Si no hay conexión a internet
	3	No tiene libros creados
Prioridad	*Alta*	
Comentarios	-	

Tabla Caso de uso: C07

RF14-1|CU14-1: El escritor podrá crear un libro.

Código	**RF14-1**
Dependencias	RF07: El escritor tendrá acceso a una lista de sus libros.
Descripción	El escritor podrá crear un libro que será añadido a la lista de sus libros, rellenando un pequeño formulario que incluya título, descripción y un cuadro donde adjuntar la imagen de la portada.
Datos específicos	No hay ningún campo obligatorio a rellenar dentro del formulario de creación.
Prioridad	Crítica.
Comentarios	Una vez se cree el libro se direccionará a la pantalla de edición del libro, donde se podrán disponibles otras funcionalidades

Tabla requisito: RF14-1

Requisito básico y crucial para la aplicación. Este requisito se hace visible desde el propio perfil del escritor, donde en la propia lista de sus libros habrá una opción clara y visible para crear nuevo libro. Al momento de pinchar en el botón, se mostrará un formulario que tendrá los siguientes campos:

1. Título: de tipo texto. Tamaño máximo de 250 caracteres.
2. Descripción: de tipo texto. Tamaño máximo de 1000 caracteres
3. Portada: de tipo texto. Representa una imagen, se almacena la ruta local de la imagen que se sube en el servidor para posteriormente cargarla.
4. Género: de tipo texto, se dará a elegir en un desplegable.

Todos los campos serán optativos.

Código	**CU14-1**	
Requisitos asociados	RF14: El escritor podrá crear un libro.	
Descripción	El escritor podrá crear un libro que será añadido a la lista de sus libros, rellenando un pequeño formulario que incluya título, descripción, género, y portada.	
Objetivo	Creación de un nuevo libro.	
Precondición	El usuario debe estar registrado y visualizando su lista de libros en su perfil.	
Secuencia normal	**Paso**	**Acción**
	1	El escritor selecciona el botón ‘nuevo libro’.
	2	El escritor visualiza una ventana de formulario de creación con título, descripción, género y portada.
	3	El escritor rellena los campos que desee.
	4	El escritor selecciona el botón de aceptar en la ventana.
	5	El escritor visualiza la pantalla de detalles del nuevo libro a la que es redireccionado.
Postcondición	Se crea un nuevo libro en la lista de libros del autor.	
Excepciones	**Paso**	**Excepción**
	1,2,3	Falla la conexión con la base de datos y no se admiten obtención, creación o modificación de datos.
Prioridad	*Crítica*	
Comentarios	-	

Tabla de caso de uso: C14-1

RF14-2|CU14-2: El escritor podrá borrar un libro.

Código	**RF14-2**
Dependencias	RF14-1: El escritor podrá crear un libro.
Descripción	El escritor podrá borrar un libro de su lista de libros que previamente haya creado. Esta opción estará disponible desde la propia lista de libros en la pantalla del perfil, y desde la pestaña de detalles de libros.
Datos específicos	Antes de borrar, se debe lanzar una alerta preguntando al usuario si está seguro de la acción, pues es irreversible.
Prioridad	Baja.
Comentarios	-

Tabla de requisito: RF14-2

El borrado de elementos es otra de las funcionales básicas que debe tener todo sistema de almacenado. El color rojo debe ser identificativo en este requisito, en caso de que se usen iconos o metáforas.

Código	**CU14-2**	
Requisitos asociados	RF14-2: El escritor podrá borrar un libro.	
Descripción	El escritor podrá borrar un libro de su lista de libros que previamente haya creado.	
Objetivo	Borrado de libro	
Precondición	El usuario debe estar visualizando su lista de libros con al menos un libro creado.	
Secuencia normal	**Paso**	**Acción**
	1	El escritor selecciona la opción borrar.
	2	El escritor visualiza una ventana de confirmación.
	3	El escritor confirma el borrado.
	4	El escritor observa que su libro ha sido borrado de la lista inicial.
Postcondición	Se borra el libro en la lista de libros del autor.	
Excepciones	**Paso**	**Excepción**
	3, 4	Falla la conexión con la base de datos y no se admiten obtención, creación o modificación de datos.
Prioridad	*Crítica*	
Comentarios	-	

Tabla de caso de uso: CU14-2

RF14-3|CU14-3: El escritor podrá editar un libro

Código	**RF14-3**
Dependencias	RF14: El escritor podrá crear un libro.
Descripción	El escritor podrá editar un libro de su lista de libros que previamente haya creado. Esta opción estará disponible en una pestaña de detalles.
Datos específicos	La edición se hará sobre el título, descripción, portada y género del libro.
Prioridad	Media
Comentarios	-

Tabla de requisito: RF14-3

Solo con respecto al libro, el escritor podrá:

- Cambiar la portada de su libro seleccionando algún otro archivo imagen local.
- Cambiar el título de su libro.
- Cambiar la descripción.
- Cambiar el género.

Código	**CU14-3**	
Requisitos asociados	RF14-3: El escritor podrá editar un libro.	
Descripción	El escritor podrá editar un libro de su lista de libros que previamente haya creado.	
Objetivo	Editar el libro	
Precondición	El usuario debe estar visualizando su lista de libros con al menos un libro creado.	
Secuencia normal	**Paso**	**Acción**
	1	El escritor selecciona un libro.
	2	El escritor visualiza una pantalla de detalles del libro.
	3	El escritor edita los detalles del libro y selecciona un botón de confirmar.
	4	El escritor observa que su libro ha sido actualizado con la nueva información.
Postcondición	Se actualiza la información del libro en la lista de libros del autor.	
Excepciones	**Paso**	**Excepción**
	2, 3, 4	Falla la conexión con la base de datos y no se admiten obtención, creación o modificación de datos.
Prioridad	*Media*	
Comentarios	-	

Tabla de caso de uso: CU14-3

RF15|CU15: El escritor podrá acceder a un libro creado y ver un listado de sus capítulos

Código	**RF15**
Dependencias	RF14-1: El escritor podrá crear un libro.
Descripción	Una vez se crea un libro, se puede acceder a sus detalles. Dentro de la pestaña de detalles, el usuario tendrá disponible la lista de los capítulos del libro en específico.
Datos específicos	Los capítulos deben mostrar título, número de palabras y fecha de última actualización.
Prioridad	Alta
Comentarios	

Tabla requisito: RF15

Cada capítulo debe mostrar:

- Título: el escritor puede poner un nombre o no, en caso de no decidir poner título, solo se incluirá numeración.
- Número de palabras del capítulo.
- Fecha de última actualización.

Código	**CU15**	
Requisitos asociados	RF15: El escritor podrá acceder a un libro creado y ver un listado de sus capítulos	
Descripción	El escritor podrá visualizar la lista de los capítulos asociada a un libro que haya creado previamente. Esta opción aparecerá en la pantalla de detalles de libros.	
Objetivo	Visualización de capítulos.	
Precondición	El usuario tiene un libro creado.	
Secuencia normal	**Paso**	**Acción**
	1	El escritor selecciona un libro de su lista.
	2	El escritor visualiza la pantalla de detalles del libro seleccionado.
	3	El escritor selecciona el apartado “Capítulos”
	4	El escritor visualiza los capítulos del libro.
Postcondición	Se visualiza una lista de los capítulos creados del libro, o en caso de que no haya, un mensaje indicándolo.	
Excepciones	**Paso**	**Excepción**
	2, 3, 4	Falla la conexión con la base de datos y no se admiten obtención, creación o modificación de datos.
Prioridad	*Alta*	
Comentarios	-	

Tabla de caso de uso: CU15

RF16-1|CU16-1: El escritor podrá crear un capítulo.

Código	**RF16-1**
Dependencias	RF15: El escritor tendrá acceso a una lista de sus capítulos.
Descripción	El escritor podrá crear un capítulo que será añadido a la lista de sus capítulos. Una vez se cree, se debe redireccionar a la pantalla de escritura.
Datos específicos	No se abre formulario.
Prioridad	Crítica
Comentarios	A raíz de este requisito se da acceso la funcionalidad de escritura en línea, uno de los gruesos de la aplicación.

Tabla requisito: RF16-1

Código	**CU16-1**	
Requisitos asociados	RF16-1: El escritor podrá crear un capítulo.	
Descripción	El escritor podrá crear un capítulo que será añadido a la lista de sus capítulos. Una vez se cree, se debe redireccionar a la pantalla de escritura.	
Objetivo	Creación de capítulo.	
Precondición	El usuario está visualizando la lista de capítulos asociados a un libro.	
Secuencia normal	**Paso**	**Acción**
	1	El escritor selecciona el botón de creación de capítulo.
	2	El escritor es redireccionado a la pantalla de escritura en línea del nuevo capítulo.
Postcondición	Se añade un nuevo capítulo a la lista de capítulos.	
Excepciones	**Paso**	**Excepción**
	2	Falla la conexión con la base de datos y no se admiten obtención, creación o modificación de datos.
Prioridad	*Crítica*	
Comentarios	-	

Tabla de caso de uso: CU16-1

RF16-2|CU16-2: El escritor podrá borrar un capítulo.

Código	**RF16-2**
Dependencias	RF16: El escritor podrá crear un capítulo.
Descripción	El escritor podrá borrar un capítulo de su lista de capítulos que previamente haya creado. Esta opción estará disponible desde la propia lista de capítulos en la pantalla de detalles del libro.
Datos específicos	Antes de borrar, se debe lanzar una alerta preguntando al usuario si está seguro de la acción, pues es irreversible.
Prioridad	Baja.
Comentarios	-

Tabla requisito: RF16-2

Código	**CU16-2**	
Requisitos asociados	RF16-2: El escritor podrá borrar un capítulo.	
Descripción	El escritor podrá borrar un capítulo de su lista de capítulos que previamente haya creado. Esta opción estará disponible desde la propia lista de capítulos en la pantalla de detalles del libro.	
Objetivo	Borrado de capítulo.	
Precondición	El usuario está visualizando la lista de capítulos asociados a un libro, con al menos un capítulo creado.	
Secuencia normal	**Paso**	**Acción**
	1	El escritor selecciona el botón de borrado de capítulo.
	2	El escritor visualiza una ventana de confirmación de borrado.
	3	El escritor confirma el borrado.
	4	El escritor observa que el capítulo se ha eliminado de la lista de capítulos inicial.
Postcondición	Se elimina el capítulo de la lista de capítulos.	
Excepciones	**Paso**	**Excepción**
	4	Falla la conexión con la base de datos y no se admiten obtención, creación o modificación de datos.
Prioridad	*Baja*	
Comentarios	-	

Tabla de caso de uso: CU16-2

RF17|CU17: El escritor tendrá acceso a una pantalla de escritura en línea.

Código	**RF17**
Dependencias	RF16: El escritor podrá crear un capítulo.
Descripción	La escritura en línea presenta un editor de texto creado específicamente para la escritura de capítulos. Una vez se cree un capítulo, se puede acceder a la escritura y se cargará el editor de texto.
Datos específicos	El editor de texto debe ocupar casi la pantalla entera, al menos un 80% de ella, para presentar mayor claridad.
Prioridad	Crítica.
Comentarios	Esta funcionalidad es clave para la aplicación.

Tabla requisito: RF17

El editor de texto que se usará para escribir será fácil e intuitivo, con colores planos y poco dañinos a la vista, y estará provisto de las opciones esenciales de fuentes, tamaños, estilos, y disposición de las letras.

Código	**CU17**	
Requisitos asociados	RF17: El escritor tendrá acceso a una pantalla de escritura en línea.	
Descripción	La escritura en línea presenta un editor de texto creado específicamente para la escritura de capítulos. Una vez se cree un capítulo, se puede acceder a la escritura y se cargará el editor de texto.	
Objetivo	Visualizar editor de texto de un capítulo.	
Precondición	El usuario está visualizando la lista de capítulos asociados a un libro.	
Secuencia normal	**Paso**	**Acción**
	1	El escritor selecciona un capítulo.
	2	El escritor es redireccionado a la pestaña de escritura en línea y visualiza el editor de texto con el contenido del capítulo cargado.
Secuencia alternativa	**Paso**	**Acción**
	1	El escritor crea un capítulo.
	2	El escritor es redireccionado a la pestaña de escritura en línea y visualiza el editor de texto con el contenido vacío.
Postcondición	Visualización del editor de texto.	
Excepciones	**Paso**	**Excepción**
	4	Falla la conexión con la base de datos y no se admiten obtención, creación o modificación de datos.
Prioridad	*Baja*	
Comentarios	-	

Tabla de caso de uso: CU17

RF18|CU18: El escritor podrá editar capítulos en un editor de texto.

Código	**RF18**
Dependencias	RF17: El escritor tendrá acceso a una pantalla de escritura en línea
Descripción	El escritor podrá editar un capítulo que previamente ha sido añadido a la lista de sus capítulos. Dicho de otra forma, esta es la funcionalidad de escribir.
Datos específicos	-
Prioridad	Crítica
Comentarios	A raíz de este requisito se da acceso la funcionalidad de escritura en línea, uno de los gruesos de la aplicación.

Tabla requisito: RF18

Código	**CU18**	
Requisitos asociados	RF18: El escritor podrá editar capítulos en un editor de texto.	
Descripción	El escritor podrá editar un capítulo que previamente ha sido añadido a la lista de sus capítulos. Esta es la funcionalidad de escribir.	
Objetivo	Escribir un capítulo.	
Precondición	El usuario está visualizando el editor de texto.	
Secuencia normal	**Paso**	**Acción**
	1	El escritor selecciona el cuerpo del editor de texto.
	2	El escritor realiza los cambios que desee sobre el texto.
	3	El escritor selecciona el botón de guardar.
Postcondición	El capítulo mantiene los cambios introducidos.	
Excepciones	**Paso**	**Excepción**
	2,3	Falla la conexión con la base de datos y no se admiten obtención, creación o modificación de datos.
Prioridad	*Crítica*	
Comentarios	-	

Tabla de caso de uso: CU18

RF19|CU19: El escritor podrá elegir opciones de tamaños, estilos y alineación en el editor de texto.

Código	**RF19**
Dependencias	RF18: El escritor podrá editar capítulos en un editor de texto.
Descripción	El escritor podrá seleccionar el tamaño, la alineación y la tipografía del texto en el que está escribiendo. Además, podrá seleccionar si lo quiere en negrita, cursiva o subrayado. Estas opciones aportan un aspecto personalizable y necesario para el
Datos específicos	Tamaño: pequeño, mediano, grande Tipografía: Sans-serif, serif. Alineación: izquierda, centro, derecha o ajustable.
Prioridad	Media
Comentarios	-

Tabla requisito: RF20

Código	**CU19**	
Requisitos asociados	RF19: El escritor podrá elegir opciones de tamaños, estilos y alineación en el editor de texto.	
Descripción	Interacción realizada entre el escritor y el editor de texto para realizar sus ajustes y dar la apariencia que desea al texto que está escribiendo.	
Objetivo	Personalizar texto.	
Precondición	El escritor está escribiendo un capítulo.	
Secuencia normal	**Paso**	**Acción**
	1	El escritor selecciona cualquiera de las opciones de estilo en la cabecera del editor.
	2	El escritor visualiza los cambios de estilo sobre su texto en el cuerpo del editor.
Postcondición	El texto cambia su formato.	
Excepciones	**Paso**	**Excepción**
	1,2	Falla la carga de Quill
Prioridad	*Media*	
Comentarios	-	

Tabla de caso de uso: CU19

RF20|CU20: El escritor podrá exportar su libro a formato Word.

Código	**RF20**
Dependencias	RF14-1: El escritor podrá crear un libro
Descripción	Con el fin de mejorar la experiencia y comodidad del escritor, este tendrá la posibilidad de exportar a formato Word un libro que previamente haya creado. La exportación recopilará la información de todos sus capítulos y la aunará en un único
Datos específicos	-
Prioridad	Baja
Comentarios	-

Tabla requisito: RF20

Código	**CU20**	
Requisitos asociados	RF20: El escritor podrá exportar su libro a formato Word.	
Descripción	Con el fin de mejorar la experiencia y comodidad del escritor, este tendrá la posibilidad de exportar a formato Word un libro que previamente haya creado. La exportación recopilará la información de todos sus capítulos y la aunará en un único	
Objetivo	Exportar libro.	
Precondición	Es escritor ha creado un libro y está visualizando sus detalles.	
Secuencia normal	**Paso**	**Acción**
	1	El escritor selecciona el botón 'exportar libro' dentro de los detalles del libro.
	2	El escritor visualiza que comienza una descarga.
	3	El escritor abre el documento descargado.
	4	El escritor visualiza el título de sus capítulos y el contenido de cada uno de ellos.
Postcondición	Se descarga un documento Word.	
Excepciones	**Paso**	**Excepción**
	2	Falla la conexión con la base de datos y no se admiten obtención, creación o modificación de datos.
Prioridad	*Baja*	
Comentarios	-	

Tabla de caso de uso: CU20

RF21|CU21: El escritor podrá hacer una corrección de ortografía de sus capítulos.

Código	**RF21**
Dependencias	RF16: El escritor podrá crear o modificar un capítulo RF18: El escritor podrá editar o escribir un capítulo.
Descripción	El escritor contará con la herramienta de corrección de texto que mostrará las faltas de ortografía de un capítulo en una lista. El escritor podrá seleccionar las sugerencias de corrección para cada falta.
Datos específicos	-
Prioridad	Alta
Comentarios	-

Tabla requisito: RF21

Código	**CU21**	
Requisitos asociados	RF21: El escritor podrá hacer una corrección de ortografía de sus capítulos.	
Descripción	Corrección de las faltas de ortografía tras la realización de un análisis ortográfico.	
Objetivo	Corregir faltas de ortografía.	
Precondición	El usuario está visualizando la pantalla de escritura en línea.	
Secuencia normal	**Paso**	**Acción**
	1	El escritor selecciona “Analizar ortografía”
	2	El escritor visualiza las faltas de ortografía que detecta el análisis.
	3	El escritor selecciona una de las sugerencias de corrección a la falta de ortografía.
	4	El escritor observa cómo el texto del capítulo dentro del editor de texto se modifica.
Postcondición	El texto es modificado.	
Excepciones	**Paso**	**Excepción**
	1,2,3	Falla la conexión con la base de datos y no se admiten obtención, creación o modificación de datos.
Prioridad	*Media*	
Comentarios	-	

Tabla de caso de uso: CU21

RF22|CU22: El escritor podrá visualizar en una gráfica el análisis de ortografía de un libro completo.

Código	**RF22**
Dependencias	RF15: El escritor podrá crear modificar o borrar un libro.
Descripción	El escritor podrá visualizar el análisis de ortografía en una gráfica de faltas por capítulos, de los libros que haya creado.
Datos específicos	En el eje X se visualiza las faltas. En el eje Y los capítulos.
Prioridad	Media
Comentarios	-

Tabla requisito: RF22

Código	**CU22**	
Requisitos asociados	RF22: El escritor podrá visualizar en una gráfica el análisis de ortografía de un libro completo.	
Descripción	Análisis de ortografía plasmado en una gráfica informativa para el escritor.	
Objetivo	Mostrar resultado de análisis en una gráfica.	
Precondición	El usuario está visualizando la pantalla de análisis ortográfico.	
Secuencia normal	**Paso**	**Acción**
	1	El escritor selecciona el libro que quiere analizar.
	2	El escritor selecciona “Analizar”.
	3	El escritor visualiza una gráfica de las faltas por capítulos.
Postcondición	La gráfica se almacena en base de datos y guarda su estado para el próximo análisis	
Excepciones	**Paso**	**Excepción**
	2,3	Falla la conexión con la base de datos y no se admiten obtención, creación o modificación de datos.
Prioridad	*Media*	
Comentarios	-	

Tabla de caso de uso: CU22

RF23|CU23: El escritor dispondrá de un guion rápido en cada capítulo.

Código	**RF23**
Dependencias	RF17: El escritor tendrá acceso a una pantalla de escritura en línea
Descripción	El escritor podrá crear su propio guion rápido, con los sucesos que irán apareciendo en el capítulo. De esta manera obtendrá una mejor construcción escénica. No debe dificultar la escritura en el editor de texto.
Datos específicos	-
Prioridad	Baja
Comentarios	-

Tabla requisito: RF23

Código	**CU23**	
Requisitos asociados	RF23: El escritor dispondrá de un guion rápido en cada capítulo.	
Descripción	Creación de un guion rápido que muestre los sucesos del capítulo.	
Objetivo	Crear un guion rápido.	
Precondición	El usuario está visualizando el editor de texto.	
Secuencia normal	**Paso**	**Acción**
	1	El escritor introduce en una caja de texto la descripción de un suceso pendiente a desarrollar en el capítulo
	2	El escritor selecciona el símbolo '+' de añadir.
	3	El visualiza que el suceso se añade a una lista y tiene una casilla de verificación asociada, y vacía.
	4	El escritor selecciona la casilla de verificación cuando crea que el suceso ya ha sido desarrollado en el capítulo.
Postcondición	El guion rápido guarda los sucesos y su estado en la lista.	
Excepciones	**Paso**	**Excepción**
	2,3,4	Falla la conexión con la base de datos y no se admiten obtención, creación o modificación de datos.
Prioridad	*Baja*	
Comentarios	-	

Tabla de caso de uso: CU23

RF26|CU26: El escritor podrá subir y visualizar imágenes de su portada, perfil, y personajes.

Código	**RF26**
Dependencias	RF04: El escritor tendrá acceso a un perfil propio. RF13: El escritor podrá crear, editar y borrar sus hojas de personajes
Descripción	El escritor podrá realizar subida de imágenes a la aplicación y visualizarlas posteriormente tanto para su perfil, portada de libros o personajes.
Datos específicos	-
Prioridad	Media
Comentarios	-

Tabla requisito: RF26

Código	**CU26**	
Requisitos asociados	RF26: El escritor podrá subir y visualizar imágenes de su portada, perfil, y personajes.	
Descripción	Realizar subida de imágenes en los formularios de creación o en la edición de detalles de libros, personajes y perfil del autor.	
Objetivo	Subida de imágenes y visualización en la aplicación.	
Precondición	El usuario está registrado.	
Secuencia normal	**Paso**	**Acción**
	1	El escritor selecciona el botón de “Subir imagen”
	2	El escritor visualiza el explorador de archivos de su sistema operativo.
	3	El escritor selecciona la imagen que quiere subir.
	4	El escritor visualiza la imagen en el espacio de imagen.
Secuencia Alternativa	Esta funcionalidad se repite en la creación de libros, en la visualización del perfil, y en la creación de personajes.	
Postcondición	La imagen se guarda y visualiza en la aplicación	
Excepciones	**Paso**	**Excepción**
	4	Falla la conexión con la base de datos y no se admiten obtención, creación o modificación de datos.
Prioridad	*Media*	
Comentarios	-	

Tabla de caso de uso: CU26

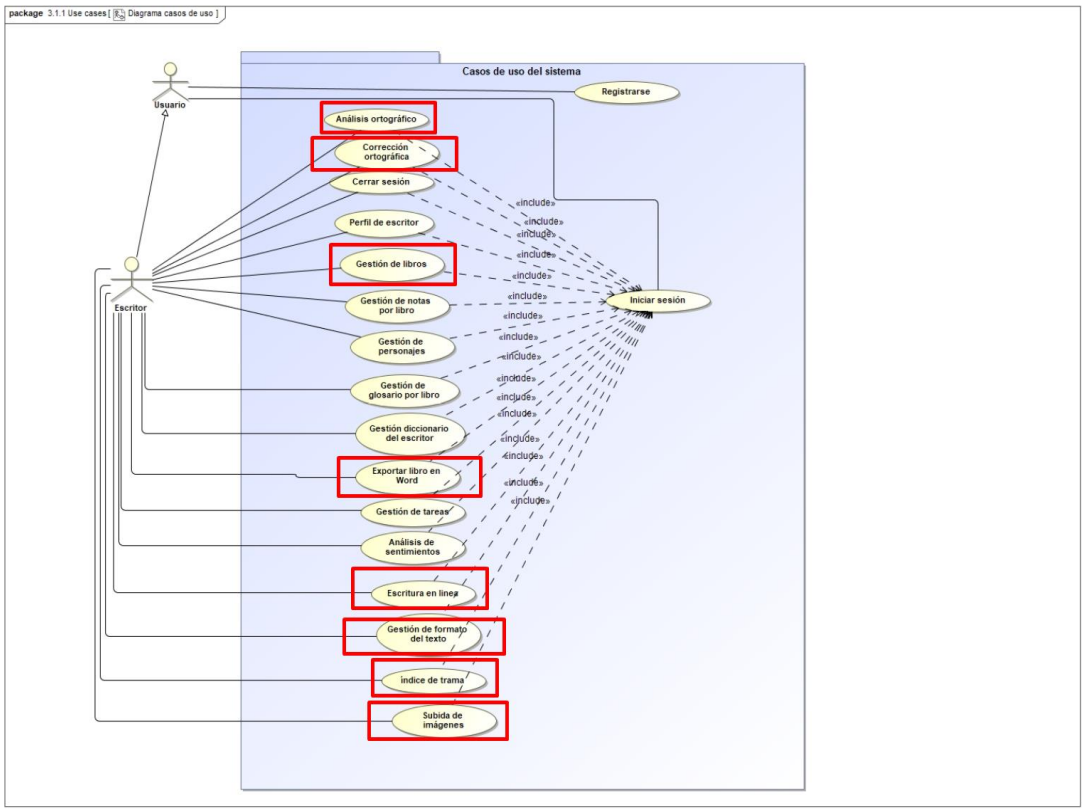


Figura 1 – Casos de uso del sistema

Se hace uso de MagicDraw como programa para crear los diferentes diagramas de requisitos y casos de uso del sistema. En las imágenes se recopila la información del sistema completo, para así tener una visión más clara y global del él, aunque las tareas realizadas para la parte que corresponde a este documento son las señaladas en rojo.

Se muestra la acción entre el usuario no registrado, y el escritor, con el sistema.

Se han definido todos los objetivos que se querían abordar desde un primer momento y que son los tratados en este proyecto, al igual que todos los requisitos funcionales derivados de dichos objetivos y, todos los casos de uso. Para ello, se hace uso de unas matrices de dependencias que cruzan objetivos y requisitos funcionales; requisitos funcionales y casos de uso, y objetivos con casos de uso.

Se han cruzado y marcado obteniendo la trazabilidad completa desde el objetivo hasta el caso de uso que lo resuelve, pasando por el requisito funcional.

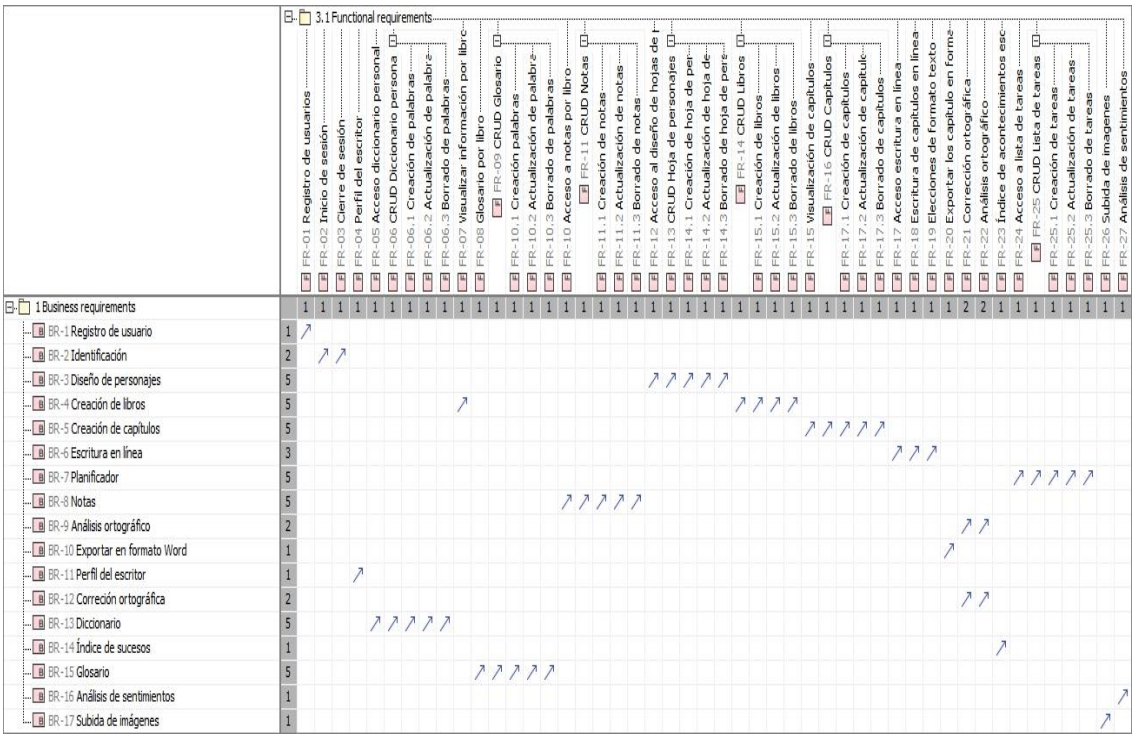


Figura 2 – Matriz de dependencia Objetivos – Requisitos funcionales

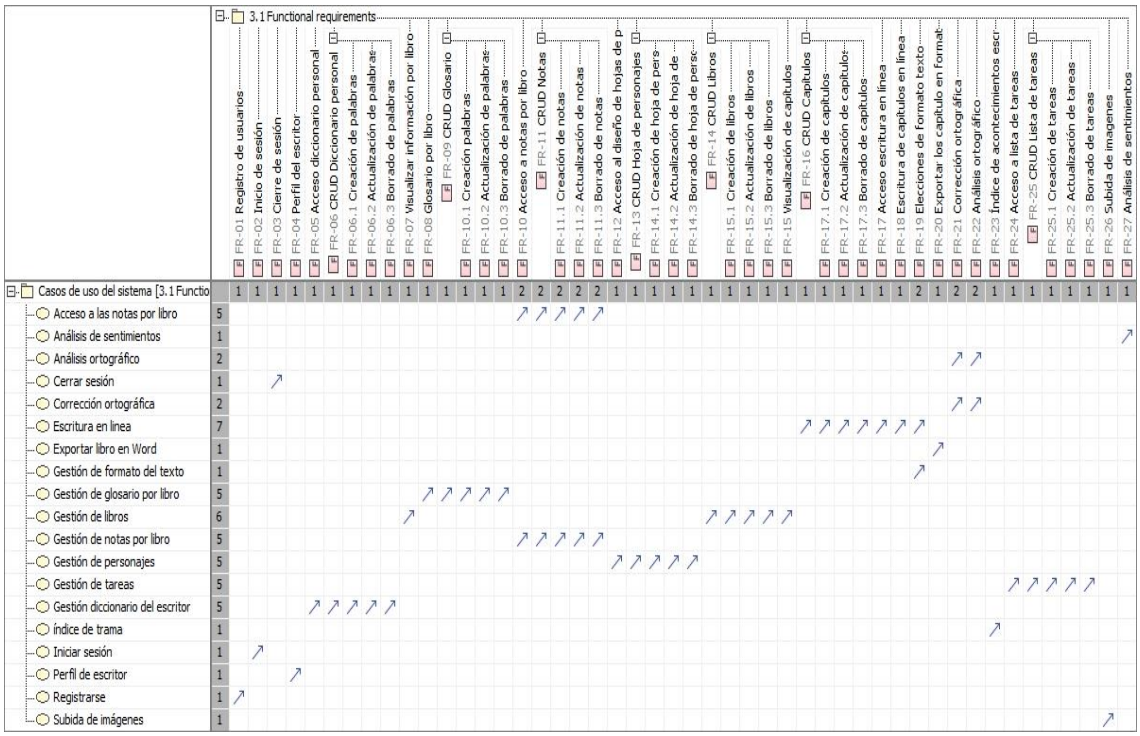


Figura 3 – Matriz de dependencia Requisitos funcionales – Casos de uso

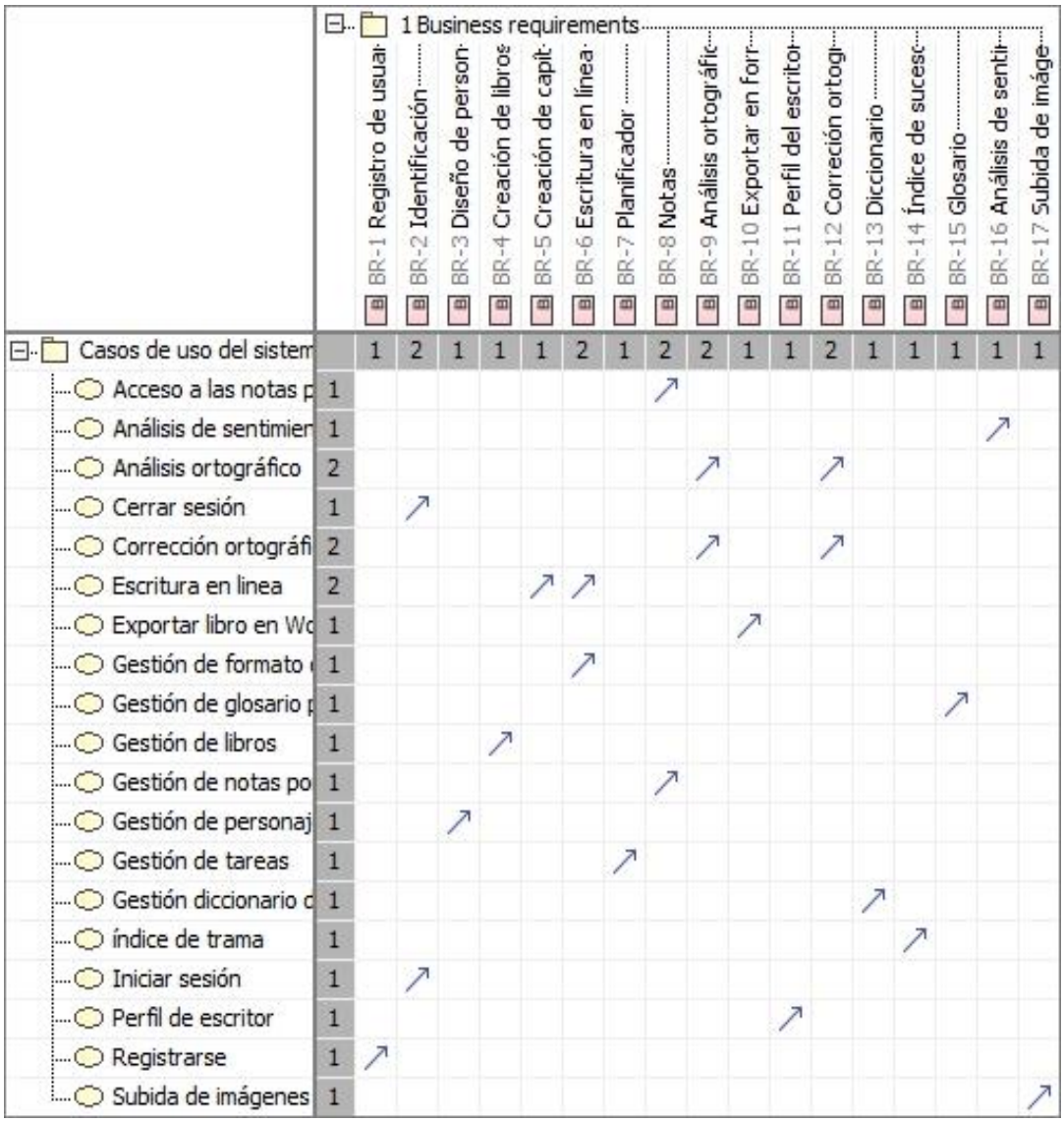


Figura 4 – Matriz de dependencia Objetivos – Casos de uso

3

Metodología

La realización de esta plataforma para escritores ha tratado de seguir una metodología Scrum[1], caracterizada por el desarrollo ágil mediante *sprints.* Al ser un proyecto realizado por dos personas, ha cobrado más sentido y realismo utilizar este método. Para ello, se ha hecho uso de la plataforma Azure DevOps[2], que ofrece un soporte para el control de versiones además de un panel Kanban con el que planificar, asignar y manejar las tareas de cada integrante. Esta opción ha sido elegida frente a otras como Jira o Trello, por el mero hecho de facilitar integración con proyectos de tecnologías Microsoft, y de una interfaz con la que se acabó familiarizando pronto

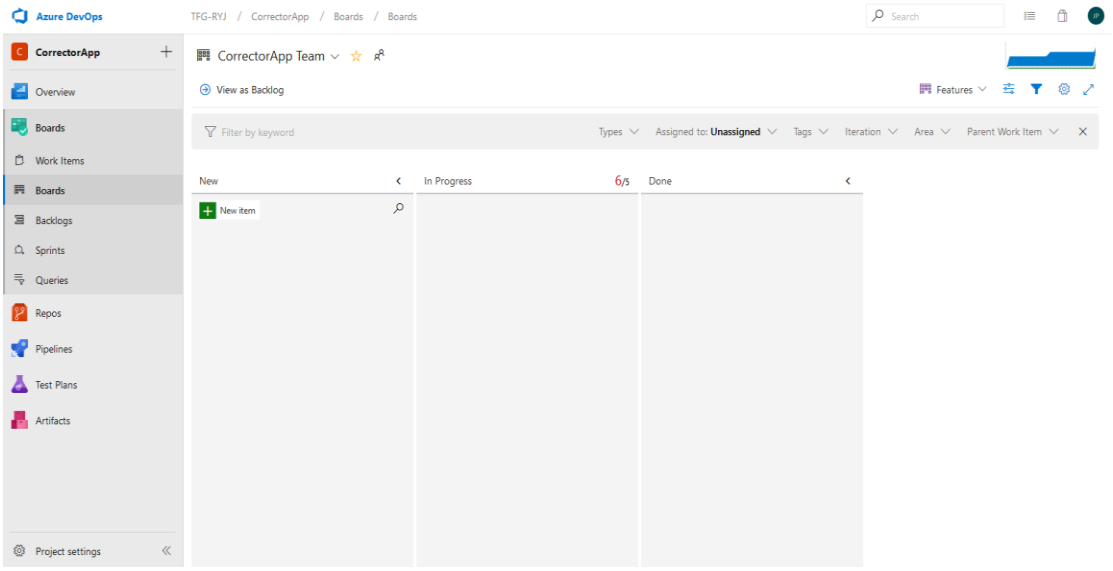


Figura 5 – Azure DevOps Kanban

Planteamiento

En primer lugar, se plantean y se desglosan las tareas referentes a los requisitos funcionales. Estas podrán ser a su vez divididas en las subtareas, de manera que el desarrollo se ajuste a la franja temporal oportuna.

Haciendo uso de la metodología Scrum, se fija el sprint en dos semanas de desarrollo, en las cuales, cada integrante del equipo debe seleccionar la tarea y subdividirla con el fin de ser lo más realista posible. Esto asignación es libre al no contar con un Product Owner.

En este momento se hace una reunión en la que se tienen en cuenta la opinión de ambos integrantes del grupo, que debaten sobre los puntos que presentan más dificultad y podrían ralentizar, así como las soluciones con las que se podría subsanar.

Tras esto, se hace uso del panel de Azure DevOps, actualizando el estado de las tareas asignadas, moviéndolas a estado de desarrollo. En el caso toparse con conflictos se procede a hacer una temprana reestimación, con el fin de no encontrarse con posibles conflictos futuros.

Una vez el sprint finaliza, se hace una reunión donde se obtienen conclusiones.

Por último, en la fase de pruebas, se procede a corregir los fallos de código, comúnmente conocidos como bugs, dedicando también un tiempo a ellos y organizándolos en el panel del Kanban.

Paralelamente a esto, se realizan reuniones diarias en las que se acota y se informa de lo que se va a realizar ese día.

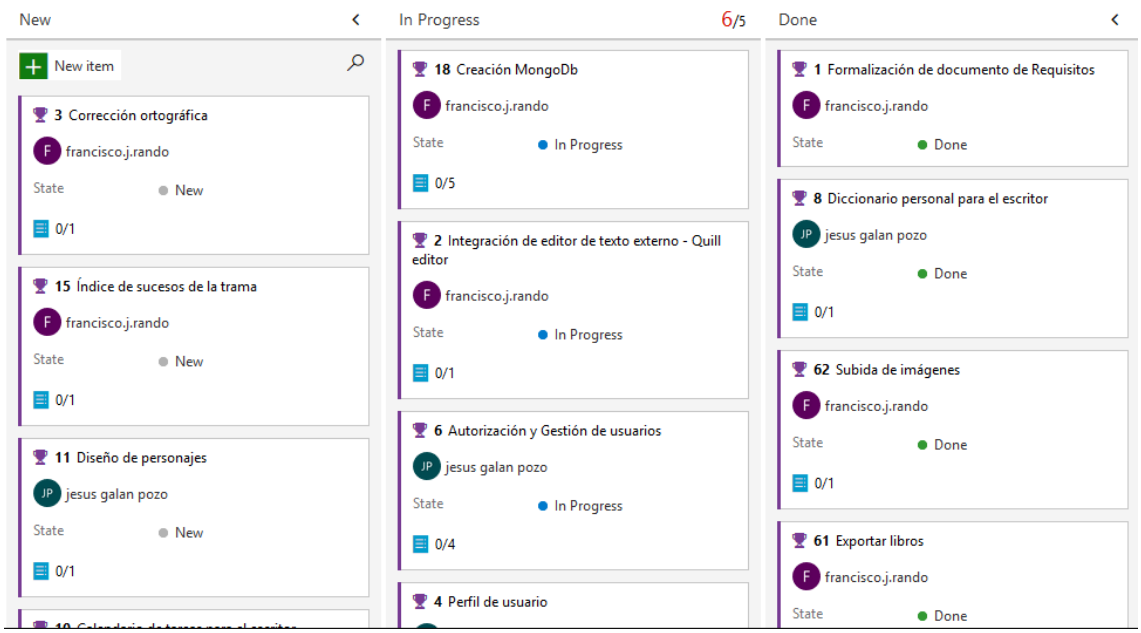


Figura 6 – Requisitos en Azure DevOps 1

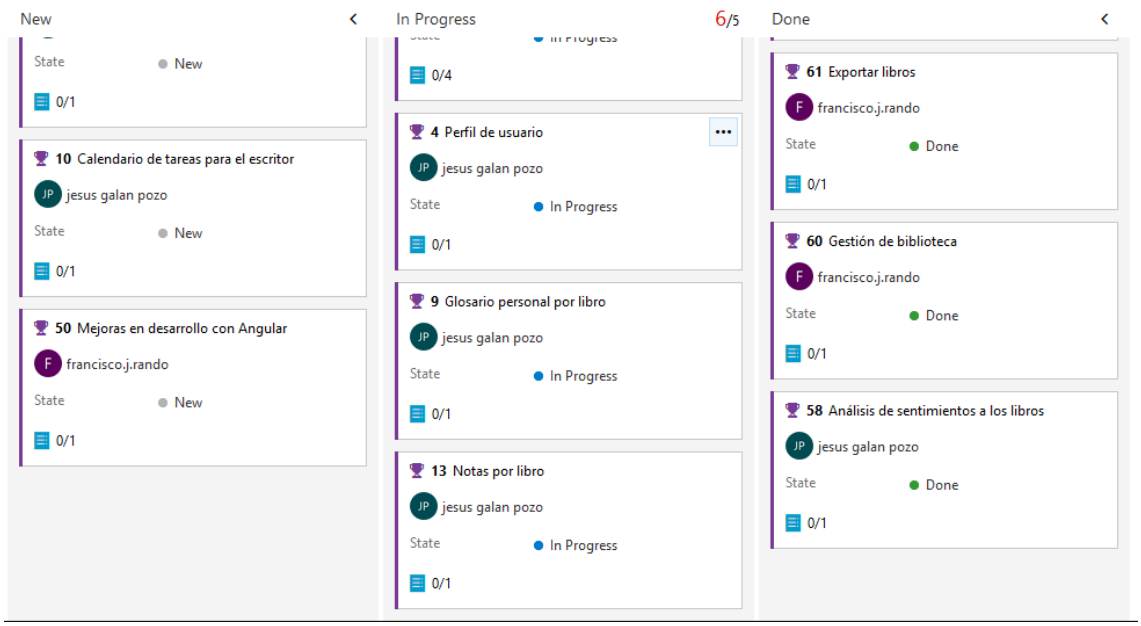


Figura 7 – Requisitos en Azure DevOps II

Experiencia

Las reuniones diarias entre los integrantes se han mantenido correctamente durante todo el desarrollo de la aplicación, y tanto el desglose como la asignación de tareas ha sido correcta. No obstante, la organización por sprints ha acabado con incumplimientos en ciertas ocasiones, debido a la imposibilidad de dedicar un tiempo fijo semanalmente.

Ambos integrantes del proyecto compaginaban la realización de este proyecto con el trabajo, por lo que la mayoría del avance se producía en fiestas, fines de semana, vacaciones y ratos libres de las tardes.

El desglose de las tareas en subtareas más pequeñas ayuda a que los conflictos, mejoras, y planes de desarrollo se consigan manejar con comunicación diaria, sin embargo, el margen fijo de dos semanas no acaba siendo suficiente en todos los casos.

4

Elección de Tecnologías

La elección de tecnologías y su uso es un pilar fundamental en el mundo de la informática, más aún con la potencia de los marcos de trabajo, la cantidad de APIs externas que se pueden incorporar y el sinfín de funcionalidades que se consiguen a un esfuerzo meramente integrativo.

En este apartado se mostrarán, una a una, las decisiones tecnológicas tomadas para llevar a cabo el proyecto.

Azure DevOps

Definición

Plataforma perteneciente a Microsoft que acompaña, facilita y unifica el flujo de vida de un software hasta la entrega del mismo. La práctica de DevOps conlleva el planteamiento y seguimiento del proyecto, desarrollo, compilación y pruebas, entrega y supervisión del producto o servicio, y operaciones. Toda esta funcionalidad se presenta en la nube.

Motivación

La elección de esta tecnología reside principalmente en la primera y segunda etapa. Por un lado, en cuanto al *planteamiento y seguimiento del proyecto*, se presenta una gran facilidad para establecer una metodología de trabajo.

El panel de tareas, más conocido como *Kanban*, potencia la organización y ayuda a visibilizar los límites del propio proyecto, y realizar iteraciones. Una metodología ágil se establece y adapta a la perfección.

En este proyecto en concreto, y teniendo en cuenta la índole grupal que este acarrea desde el principio, es indudable la aportación de esta herramienta.

En segundo lugar, la etapa del desarrollo de la aplicación está perfectamente adaptada para generar un entorno remoto y sincronizado en el que controlar los cambios del proyecto y sus versiones.

De hecho, incorpora la opción de trabajar con Git[3] y TFS, herramientas de control de versiones. En este proyecto, se decide trabajar con la primera.

Nuevamente, un trabajo grupal no podía sino exigir este beneficio.

Experiencia

En cuanto a la primera etapa, se experimenta un planteamiento óptimo y visible al que se le pueden dedicar horas. El desglose y abanico de elementos no se limita solo a definir tareas, también bugs e historias de usuario. Todas ellas clasificadas en planificadas, activas, cerradas y en pruebas.

Sin duda, es conveniente dedicar tiempo a la planificación, aunque no hay que olvidar que también existen límites.

Planificar, también es una tarea, por lo que hay que reservarle un espacio.

Y si bien la organización es favorable, el control de versiones con Git se siente realmente orgánico y ágil. Durante el proyecto, no se presentan conflictos de integración en este aspecto, y desde Visual Studio, se consigue una conexión remota sin dificultad que el trabajo grupal ha notado con creces.

Muy buena experiencia en este sentido.

Azure CosmosDB

Definición

Servicio de base de datos de Microsoft multimodelo con un enfoque NoSql. Proporciona alojamiento en la nube.

Motivación

Un alojamiento de datos común y en la nube es muy atractivo a la hora de trabajar en grupo, y presenta una gran utilidad en cuanto a portabilidad. Es única, no hay que tener replicas locales, pudiendo acceder a ellas desde cualquier dispositivo.

Otro factor fundamental es la elección de base de datos no relacional para este proyecto. En términos de escalabilidad, y flexibilidad se obtiene una gran ventaja de tiempo.

Salta a la vista que una aplicación que trata con libros, capítulos o notas, habla en términos de documentos. Para este tipo de enfoque, una base de datos documental, traducida a formato Json, proporciona agilidad, con una latencia menor a diez milisegundos.

Experiencia

MongoDB es el modelo elegido dentro de los que componen CosmosDB[4]. La interfaz de métodos de MongoDB supuso una fácil integración con Asp.net Core. Aunque es cierto que un inicio hubo dificultad de traducción a la hora de almacenar los modelos de back-end en CosmosDB. Se encontraron conflictos con la traducción de identificadores, hasta que se decidió tratarlos internamente y almacenarlos de tipo texto.

La traducción a Json[5] se tuvo que marcar en el modelo de forma explícita, con atributos de la clase BsonElement sobre las propiedades, para que se produjera la relación entre lo que se pretendía leer y lo que estaba almacenado en la nube.

Una vez se tomó la mecánica, se comenzó a experimentar la agilidad.

La interfaz y gestión de Azure sobre el modelo de datos es limpia y clara, y permite una buena comunicación y acceso entre los desarrolladores, sin mantener dependencias con servidores locales.

Las modificaciones que se tuvieron que hacer sobre el modelo no supusieron daños severos, y en cuestión de minutos, se podía continuar con el desarrollo.

Swagger

Definición

Marco de trabajo que engloba una serie de reglas, especificaciones y herramientas que facilitan el entendimiento de una API aportando una documentación de la misma.

Motivación

Las Api no mantienen un lenguaje común, y Swagger[6] aparece para hacer legibles cada API. Swagger UI presenta una interfaz que engloba cada controlador, cada medio de entrada, junto a los parámetros y las respuestas. Todo en el formato que la API entiende.

Esta interfaz anticipa errores, pues desde ella se pueden lanzar peticiones a cada uno de los controladores, ver los resultados por pantalla y actuar en función a ello.

Experiencia

Con una integración que no supuso más problema que la instalación de un paquete, y una fácil configuración en el proyecto, Swagger se presenta como una herramienta que nunca viene de más, que toda api debería incorporar pues no supone gran esfuerzo hacerlo.

Su interfaz es limpia, realmente entendible, y amena de utilizar. La visibilidad que se consigue en cada método del controlador es más que satisfactoria, y la documentación de modelos que incluye es realmente útil a la hora de transmitir conocimiento de la API.

Visual Studio 2017

Definición

Entorno de trabajo desarrollado por Microsoft, integrado para Windows, Linux y MacOS, y que proporciona compatibilidad para múltiples lenguajes como C#, Visual Basic o C++.

Motivación

Visual Studio[8] es un entorno de desarrollo que aporta todo lo que un desarrollador puede necesitar. Su versión gratuita no presenta ningún inconveniente ni deja nada que desear para cualquier requerimiento, y al mismo tiempo, sus versiones profesional y Enterprise presentan un conjunto de mejoras que facilitan el desarrollo a un gran nivel.

La documentación de Microsoft y la comunidad es extensa y no deja que el desarrollador se estanque ni tenga problemas en la integración.

La creación de proyectos, la gran personalización de interfaz, y las múltiples herramientas que presentan hacen de Visual Studio un entorno muy potente.

Apostar por las tecnologías de Microsoft significa poner Visual Studio al principio de la lista. Su integración, comunicación y adaptabilidad para el resto de las tecnologías y para los lenguajes propios de Microsoft es muy destacable.

La depuración presenta un alto nivel de opciones,

Experiencia

Visual Studio 2017 supone una alta comodidad para el desarrollo, y algunas de sus herramientas se hacen notar en este proyecto.

El gestor de versiones incorporado en el propio entorno provoca que Git sea muy cómodo de usar.

La función de depuración y sus múltiples configuraciones consiguen que los fallos sean mucho más fáciles de detectar.

Y por supuesto, a nivel de agilidad y experiencia se consigue una gran mejoría gracias a, por un lado, mecanismo *intellisense,* que permite el autocompletado de funciones, declaración de variables y propiedades, bucles o condiciones; y, por otro lado, al gestor de paquetes, las plantillas de creación de proyectos, clases, enumerados o interfaces que ponen todo a favor para que el desarrollador comience a crear.

ASP.Net core Web Api

Definición

Marco de trabajo multiplataforma de código abierto y gratuito que tiene como principal finalidad el desarrollo web, aventajándose de su predecesor ASP.net[9] por un mayor rendimiento. Desarrollado por Microsoft y la propia comunidad. Enfocado en la creación de Api para web, proporcionando un servicio modular.

Motivación

Asp.net Core Web Api presenta una serie de aportaciones que condujeron a su elección.

En primer lugar, su lenguaje C# mantiene y profundiza en los conceptos de programación orientada a objetos (OOP). La aportación de expresiones Lambda, o Linq como librería para trabajar e iterar sobre conjuntos, hacen cobrar peso a este lenguaje.

Asp.net, ya como base, proporciona un compilado, un conjunto de librerías completo y un recolector de basura, pero junto a .net core termina por completar la librería de clases con un enfoque web, que al mismo tiempo es más liviano frente al de Asp.net Framework; puesto que se centra en proporcionar un núcleo básico que puede ir ampliándose.

Si a este hecho se le suma, la programación multilenguaje (C#, visual basic, C++), el desarrollo multiplataforma, y la integración de paquetes Nuget para cubrir los distintos aspectos del desarrollo web que surjan, se consigue un entorno portable, escalable, y flexible.

Esta provisto, además, de una estructura base o esqueleto para la creación de proyectos, perfectamente preparado para la inclusión de patrones.

Uno de ellos, y el mejor conseguido, es la inyección de dependencias.

Se trata de una solución eficaz a la inversión de control, designando la instanciación al propio programador en lugar de a los constructores de las clases, especialmente diseñadas para la etapa de pruebas.

Pero, además, prepara un contexto para la construcción de API Rest[10], una interfaz de comunicación entre el lado cliente y el lado servidor, que no mantiene estado, caracterizada por ser realmente intuitiva.

Proporciona cuatro métodos: get, post, put, delete. Para obtener, modificar, enviar, y borrar datos a través de peticiones Http.

El hecho de construir una buena API permite aislar el desarrollo back-end y hacerlo reutilizable para distintos marcos de trabajo de front-end.

Experiencia

En este marco de trabajo se aprecia una experiencia de desarrollo muy intuitiva y fácil de llevar. Una documentación totalmente activa y actualizada provee al desarrollador de gran confianza y ritmo a la hora de construir nuevas funcionalidades y generar estructuras.

El gestor de paquetes Nugets[11] genera un continuo aporte de funcionalidades, que como único punto negativo es la adaptación de algunos paquetes que aún están solo disponibles para Asp.net framework. Entre ellos destacan, Swagger, Newtonsoft[12], JWT[13], MongoDB[14] driver y MongoDB[15].

Por último, mencionar que su lenguaje C#[16] resulta robusto y estable, y se sigue apostando por sus mejoras.

Angular

Definición

Marco de trabajo para aplicaciones web de código abierto, que está desarrollado en Typescript[17] y mantenido por Google. Su principal objetivo es potenciar la creación de aplicaciones en navegador, mejorando la experiencia a nivel de pruebas y desarrollo.

Motivación

Este proyecto se presenta como una oportunidad perfecta para embarcarse en la creación de aplicaciones *full-stack,* la inexperiencia previa de los desarrolladores en el desarrollo del lado cliente, provoca una gran inquietud sobre qué marco de trabajo utilizar, cuáles son los más potentes a día de hoy, y qué curva de aprendizaje presentan.

Entre los candidatos principales se barajaban Vue.js, React, y Angular[18].

El primero de ellos se descartó por su reciente aparición, y a nivel documentación y respaldo de la comunidad, presentaba un aprendizaje más dificultoso que en los otros dos.

El debate se situó pues en si elegir React o Angular, y tras debatir durante el margen de una semana, se llegó a la conclusión de que la curva de aprendizaje de Angular era más suavizada con respecto a la de React.

Angular presenta el lenguaje Typescript como una nueva vía para hacer la experiencia de desarrollo cliente mucho más visual, amena y legible. Tiene aspectos comunes con el lenguaje de programación orientado a objetos C#, como por ejemplo el tipado de variables, concepto de clases, y herencia. Además, detecta y muestra errores en tiempo de compilación.

React, en su defecto, además de presentar la dificultad de usar JavaScript como lenguaje para quien no lo haya usado antes, rompe con el esquema habitual de páginas HTML, haciendo que estén contenidas en constantes.

Experiencia

Pese a presentar un lenguaje de programación realmente familiar para los desarrolladores del lado servidor, Angular supuso todo un reto a nivel de integración y domino.

La primera dificultad encontrada fue su gran número de dependencias con otros paquetes a la hora de integrar y actualizar versiones, más aun teniendo en cuenta la velocidad con la que Angular pasa de una versión a otra superior. Cuando se empezó a realizar este proyecto, la última versión de Angular era 5, y esta fue la que se escogió.

Tras un tiempo, y con el apoyo de una buena documentación oficial y cursos, se comprendió la lógica de componentes, módulos, enrutado y comunicación entre las distintas capas desarrolladas en Angular.

Además, tras dominar Angular-cli como gestor de comandos, tanto la creación de proyectos, como la distribución de componentes se hizo más fácil y rápida, que junto a su combinación con Node.js para el acople de nuevos componentes externos, terminó por dar un gran impulso a la integración y escalabilidad.

El siguiente problema fue que, aunque se estableció una base para poder realizar el desarrollo, la jerarquía entre componentes y la distribución de módulos dejó de ser la apropiada para la envergadura que estaba tomando el proyecto.

Realizar una separación y reestructuración del esqueleto en el lado cliente supuso subir un escalón más en la curva de aprendizaje.

Tras ello, el desarrollo se agilizó en gran medida. Destacar la ventaja a nivel de integración de componentes externos[19] que aportaron funcionalidad de manera inmediata, tanto para cubrir requisitos funcionales, como para potenciar el atractivo de la aplicación. En el siguiente desglose se pueden apreciar cuales fueron los más relevantes:

Utilidad	Componente
Editor de texto	ngx-quill quill.js
Potenciar Interfaz para Angular	ngx-Bootstrap ng2-Select Bootstrap ng5-slider
Exportación libros a Word	Docx file-saver
Validación de formularios	ng2-validation
Creación de Gráficas	Charts.js
Iconos	Font-awesome

5

Diseño del modelo de datos

Como se menciona en el apartado de tecnologías, se decide escoger una base de datos no relacional, cuyo almacenamiento y acceso está combinado e integrado con una API REST, proporcionando la persistencia a la base de datos CosmosDB integrada en Azure.

Es importante remarcar que se realiza un control interno de identificadores entre los distintos modelos. Con el fin de mantener un manejo consistente, y sencillo, estos campos se crean adicionalmente al propio ID que genera la base de datos, y son identificadores únicos. Se almacenan como tipo texto, globalmente conocidos como string, pues a nivel de peticiones al controlador REST hacía más fácil el manejo; ya que realizar peticiones a la API con tipos básicos distintos a texto dificulta la construcción del URL.

Además, la relación de modelos entre Angular y ASP.net Core, presenta una ligera diferencia en ciertos tipos básicos. En el lenguaje Typescript no existe tipo int, double, o float, todos ellos se globalizan en el tipo number. Con el fin de evitar errores de transformación en ambas partes, se decide mantener consistencia en ambos modelos dejando los identificadores de tipo texto.

A partir de la herramienta Swagger, podemos extraer un esquema básico de cada modelo de datos.

Writer

Figura 8 – Modelo de writer

En este modelo se puede ver las propiedades que va a tener un escritor cuando se registre en el sistema. Esta Información básica como puede ser el email con el que se registra, contraseña, nombre completo, alias entre otras. Toda esta información no es introducida por el usuario, sino que se autogeneran ya que se requieren en ciertas partes de la plataforma para una mejor navegación a través de ella. Estos campos no introducidos por el escritor son el refreshToken y el Salt, que se utilizan para la gestión de los tokens de acceso y para añadir aleatoriedad a la contraseña para una mayor seguridad respectivamente. Adicionalmente este modelo contiene dos listas de objetos de otros modelos como son la lista de palabras del diccionario y la lista de tareas, estos dos modelos se explicarán a continuación.

DictionaryWord

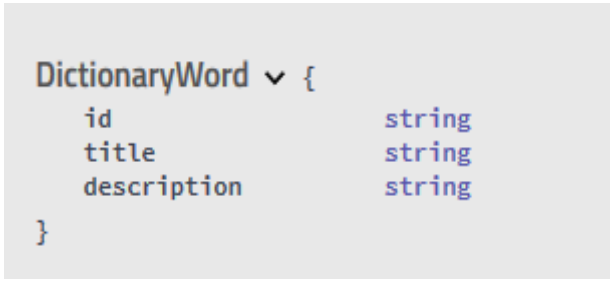


Figura 9 – Modelo de dictionaryWord

Este modelo es usado para las palabras que puede almacenar el escritor. Dicho modelo está incrustado en el modelo de *Writer* en forma de una lista como se ha podido observar.

TaskWriter

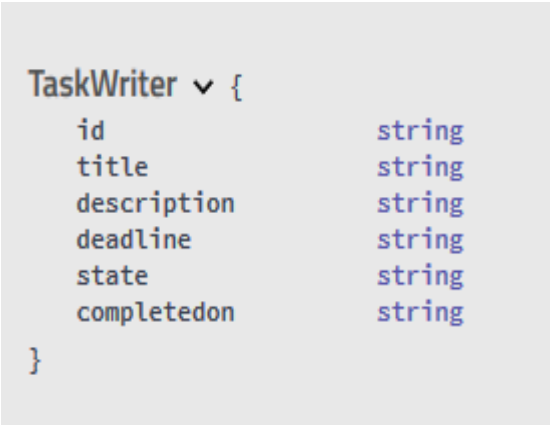


Figura 10 – Modelo de taskWriter

Este modelo hace referencia a las tareas que el usuario podrá gestionar en el apartado del planificador. Este modelo tiene las propiedades básicas y necesarias para que el usuario pueda organizarse creando tareas. Algunas de sus propiedades pueden ser el título, descripción, el estado en el que se encuentra dicha tarea, la fecha límite que el escritor indica y la fecha de cierre que se informa una vez se cierra la tarea. Este modelo está incrustado en el modelo de *Writer* en forma de lista.

Book

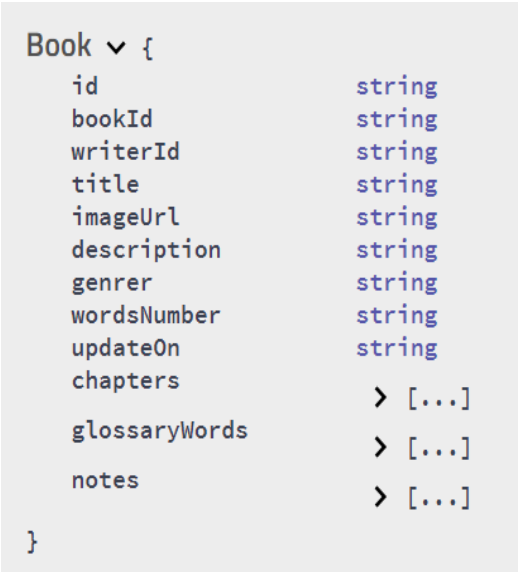


Figura 11 – Modelo de book

En este modelo se puede observar las propiedades que tendrá un libro al ser creado en el sistema. Los campos bookId, writerId, wordsNumber, updateOn, e imageUrl son generados y gestionados por el propio sistema. El campo wordsNumber representa un contador de palabras del libro completo, updateOn la fecha de actualización del libro, e imageUrl almacena la ruta de la imagen de portada. Los campos restantes de tipo string se obtienen a raíz de la inserción por parte del usuario.

Por la simplicidad que presentan y por la relación funcional que mantienen con un libro, también se incluyen tres subconjuntos en el modelo de Book: chapters, glossaryWords, y notes.

GlossaryWord

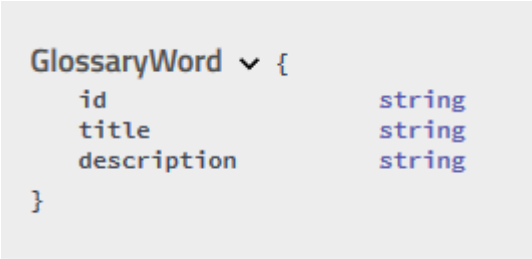


Figura 12 – Modelo de glossaryWord

Este modelo refleja las palabras que el escritor puede almacenar para explicar y dar significado propio a una serie de palabras que aparezcan en cualquiera de sus obras. Esta lista de palabras del glosario están incrustadas en el modelo de *Book*, es decir, cada libro contiene su propio glosario.

Note

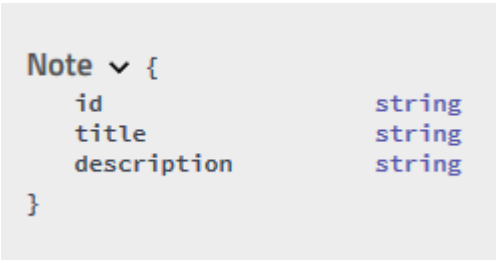


Figura 13 – Modelo de note

Este modelo refleja las notas que puede realizar el escritor en cada libro. Estas notas están compuestas por un título y una descripción donde se explicará algún suceso que decida el propio escritor. Este modelo está incrustado en forma de lista en el modelo de Book, es decir, cada libro contiene su propio glosario.

SentimentsAnalytic

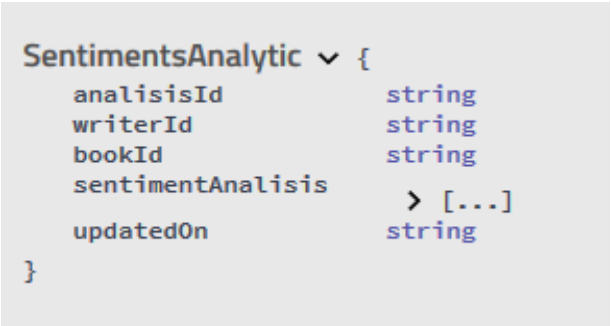


Figura 14 – Modelo de sentimentsAnalytic

Este modelo hace referencia al análisis de sentimientos que el escritor tiene la opción de realizar una vez que ha escrito algún capítulo de un libro. La respuesta a la petición del análisis se almacenará en el atributo *sentimentAnalisis* el cual contendrá toda la información relevante al análisis realizado. Las otras propiedades que se pueden observar son internas para saber a qué escritor y libro pertenece dicho análisis.

Chapter

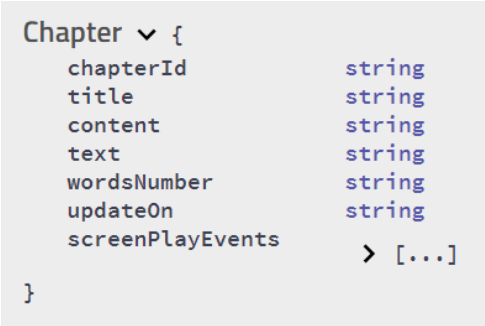


Figura 15 – Modelo de chapter

El modelo Chapter, representa el capítulo contenido en el libro. Sus campos reciben la información a partir del escritor, durante la creación de libro, o durante la propia escritura. ChapterId, wordsNumber, y updateOn son generados por el sistema.

En esta ocasión, wordsNumber representa el número de palabras del capítulo. La suma de todos los números de palabras, de todos los capítulos de un libro, genera el número de palabras contenido en wordsNumber del libro. Content y text almacenan el cuerpo del capítulo, con la diferencia de que el primero mantiene el formato del editor, y el segundo es tan solo texto plano.
Además, mantiene un subconjunto denominado screenPlayEvents, que supone ni más ni menos que los eventos del guion rápido de cada capítulo.

ScreenPlayEvent

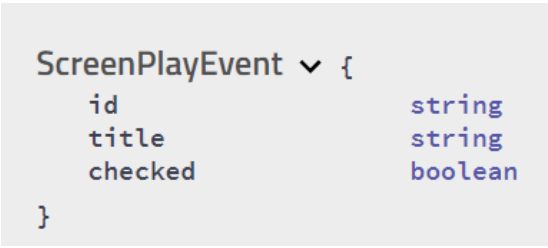


Figura 16 – Modelo de screenPlayEvent

Este modelo se alimentará cada vez que un escritor introduzca nuevos sucesos o eventos que se producen durante el desarrollo del capítulo. El campo checked hace referencia a si el evento ha sido completado a lo largo del capítulo, o bien está pendiente a realizar.

TextAnalytic

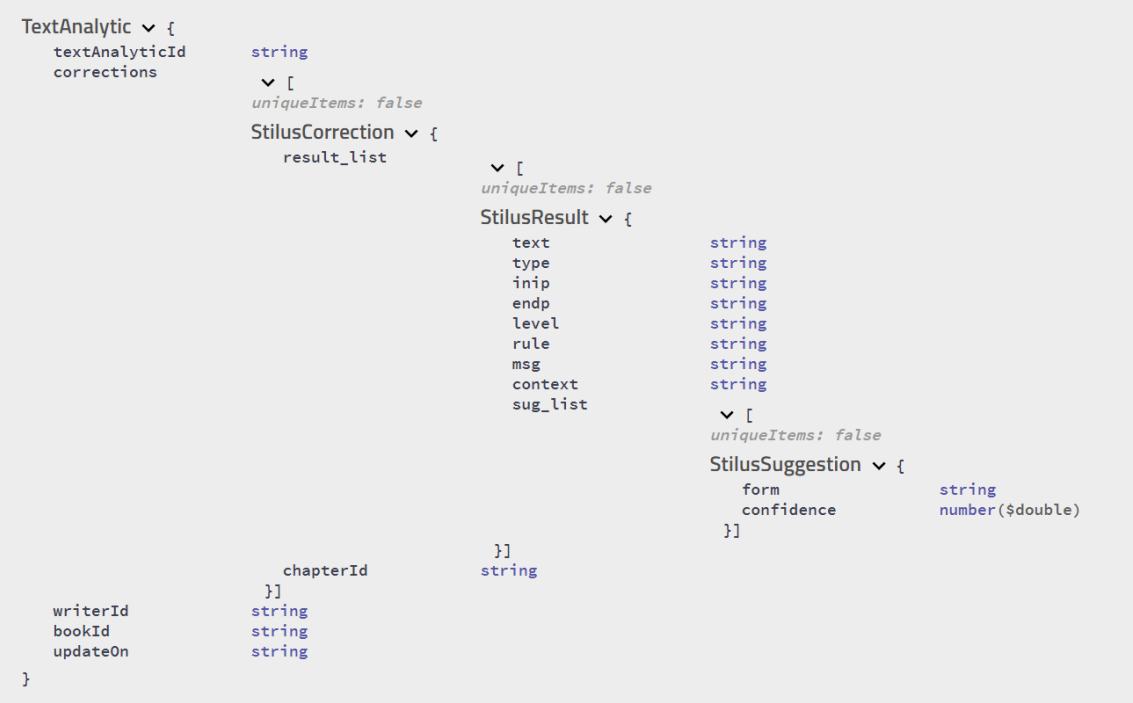


Figura 17 – Modelo de textAnalytic

En esta imagen se puede apreciar el modelo de *TextAnalytic*, almacenamiento de los análisis de texto realizados para la corrección de faltas de ortografía.

Es un modelo pensado para que cada libro tenga un solo análisis, creado una única vez, que se irá actualizando en las siguientes peticiones de análisis. Mantiene su identificador único, pero se vale de los identificadores de escritor y libro para comprobar la existencia de análisis previo.

Así mismo, el subconjunto *corrections* mantiene el identificador del capítulo del que proceden las faltas de ortografía. Este subconjunto es el mayor grueso del modelo, pues representa la respuesta generada por la API de *Stilus*, la cual dota al sistema de información realmente interesante.

Cabe destacar el texto erróneo (ya sea una frase o palabra), el mensaje explicativo del error, y el conjunto de sugerencias para resolverlo.

Character

El modelo de personajes refleja toda la información que necesita el escritor para moldear a los personajes que van a dar vida a las obras, por lo tanto, necesita almacenar una serie de propiedades y características propias de cada personaje. Todas estas propiedades las insertará el escritor para dar personalidad y forma a los personajes que el decida, y podrá consultarlos en la galería de personajes en todo momento.

Figura 18 – Modelo de carácter

6

Desarrollo de la aplicación

El desarrollo de la aplicación consta de una serie de soluciones ante requisitos, solventando los problemas que se encuentran, con el fin de cumplir un objetivo. En este apartado se expondrá pues, las soluciones dadas a cada requisito de los comentados en el apartado de especificación de requisitos y casos de uso, tratando de mostrar cuál fue la experiencia durante el desarrollo.

Acceso a lista de libros

El desarrollo de esta funcionalidad mantiene relación con la existencia de un registro por parte del usuario, puesto que los libros serán visibles en el apartado nombrado como “Mi biblioteca”, a la que solo un escritor puede tener acceso, y, por tanto, será necesario obtener el id de dicho escritor para obtener sus libros.

En esta pantalla, se decide reservar un espacio para mostrar un listado de libros. Un espacio claro, y suficientemente grande como para cobrar la importancia que merece.

En un principio, se quiso dividir la pantalla en una cabecera y un cuerpo, y en este último irían los libros. No obstante, se descubrió que el aprovechamiento de la pantalla era menor, y el protagonismo del listado de libros quedaba minimizado.

Por tanto, se decide que los libros queden ocupando el grueso derecho de la pantalla en una lista reservada.

Nada más acceder a la pantalla, el lado cliente enviará una petición Http Get transportando dicho id al controlador del lado del servidor, y este consultará en la base de datos qué libros hay para el escritor con el id que se recibe por parámetro.

Mientras esto ocurre, en pantalla se muestra una animación de carga, comúnmente conocida como *Spinner*. En el momento en el que el servidor devuelva una respuesta, el spinner desaparecerá.

Puede ocurrir que el escritor no tenga creado ningún libro aún, para este caso, la lista estará vacía y se incita al escritor a que cree su primer libro con un mensaje claro “Aún no has creado ningún libro ¿A qué esperas?”

A raíz de este acceso a la lista de libros surgen los otros requisitos relacionados con la **creación, edición y borrado** de libros, y a raíz de ellos, se va alimentando la lista.

Cuando sí que hay libros creados, cada elemento de la lista muestra:

- Portada: Si durante la creación se ha incluido portada, se muestra en un espacio a la izquierda de tamaño fijo. En caso de haber creado el libro sin portada, se muestra una imagen por defecto en el mismo espacio reservado.
- Título: En grande y arriba. Si no se ha introducido título en la creación, se muestra “[Sin título]”.
- Descripción: Justo abajo del título, tendrá un espacio de tres líneas reservado, y en caso de sobrepasarlas se corta el texto con “…”. Si no se ha introducido descripción en el formulario de creación, se muestra “[Sin descripción]”.
- Fecha última actualización: Colocada a la misma altura del título, pero desplazada a la derecha y en un tamaño menos llamativo. Siempre aparece desde que se crea el libro, pues es un parámetro interno.
- Autor: Debajo de la descripción. Siempre aparece, pues no es un parámetro de entrada del formulario.

Dentro de *NgBootstrap,* librería preparada especialmente para el desarrollo *front-end* en Angular, presenta una directiva denominada *NgbTab*, que proporciona una ayuda para crear un tabulado con el que aprovechar el espacio de la pantalla. “Detalles”, “Libros” y “Glosario”, componen este tabulado, mostrando por defecto el apartado de “Libros”.

Crear, borrar y editar libros

Este requisito se divide en tres partes, y para cada una de ellas se aporta una solución en específico.

1. **Crear libro**

En el apartado de libros, se presenta un botón “Nuevo libro” claro y visible, arriba a la derecha. Tras pulsar el botón se muestra una ventana, denominada en términos informáticos como *Modal*. Este tipo de elemento se utiliza cuando la información a mostrar no es lo suficientemente abundante como para necesitar una pantalla aparte. Es muy común verla en inicios de sesión, y se decidió optar por ella en la creación de libros.

Para ello, se recurre de nuevo a NgBootstrap, usando el servicio *NgbModal,* pues facilita el control de los modales y su configuración en el marco de trabajo Angular, además de agilizar el desarrollo y proporcionar un aspecto atractivo.

El formulario se presenta simple, a la izquierda un apartado reservado a la subida de una portada. A la derecha, los parámetros de entrada título, género y descripción.

Dentro del propio modal se presenta la opción de cerrar “x” arriba a la derecha, “Cancelar” en el pie de la ventana, y “Confirmar” también en el pie, pero de un color azul llamativo.

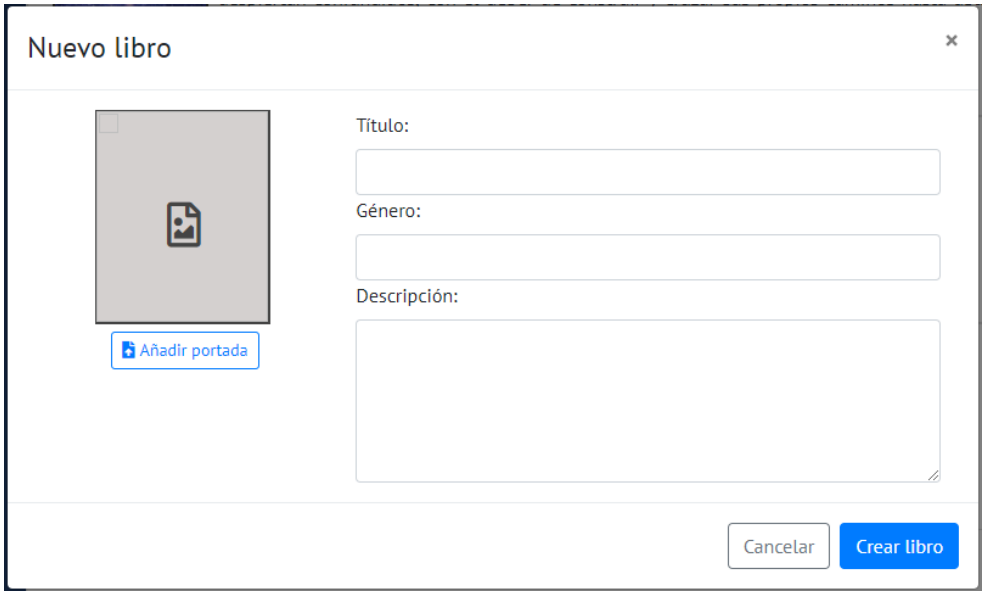


Figura 19 - Formulario creación libro

Una vez se pulse el botón de confirmar se redirecciona a la pantalla de detalles del libro, y si el escritor decide volver a su biblioteca, encontrará el nuevo libro añadido a la lista.

Puesto que la creación de un libro es muy dinámica y está realmente abierta a la futura modificación, ningún parámetro de entrada es obligatorio. Es por ello, que se redirecciona a la pantalla de detalles del libro, pues se proporciona facilidad a la modificación; dando la posibilidad de tomar la decisión de estos parámetros en cualquier momento.

De esta forma, un escritor se puede centrar en otros aspectos creativos, o directamente comenzar a escribir capítulos, aunque no sepa qué título tendrá su historia.

Durante esta interacción, el lado cliente ha obtenido el objeto libro introducido y realiza una petición Http Post al lado controlador del lado servidor, que se comunicará con la base de datos e incluirá el nuevo elemento.

2. Borrar libro

Con un libro creado en el apartado de libros basta para visualizar la funcionalidad de borrado.

En cada elemento de la lista se presenta un claro botón rojo con el símbolo de la papelera, colocado bien apartado en la esquina inferior derecha. No todos los días se quiere eliminar una creación, y por ello, además de aislar el botón, se presenta un nuevo modal al pulsarlo.

El modal de confirmación es muy típico en operaciones de borrado significativas y permanentes. En esta aplicación también se opta por ellas, de esta manera, si el escritor pulsa el botón de borrado por equivocación no se producirá el borrado instantáneo, en lugar de ello se abre la siguiente ventana:

Figura 20 - Ventana de confirmación de borrado

Cuando el escritor pulsa el botón Ok, se produce el borrado completo del libro, y este desaparecerá de la lista, pues el lado cliente obtiene el id de dicho libro y realiza la petición Http Delete al controlador propio del lado servidor. En ese instante, se localiza el Id en la base de datos y se procede a su eliminación.

3. Editar libro

La edición del libro se produce en la pantalla de detalles del libro. Esta pantalla recoge varias funcionalidades que se acometerán a lo largo del documento.

En la pantalla, se respeta la misma distribución que se sigue en “Mi biblioteca”, con una carta identificativa, conocida en términos de front-end como “card”, a la izquierda que representa la información principal del libro; y una tabulación a la derecha que expone “Detalles”, “Capítulos”, “Notas”, “Glosario”.

Por defecto, se muestra “Detalles”, un formulario con los campos de Título, Categoría y Descripción, que mantienen la información introducida en la creación del libro. El escritor podrá modificar el contenido de estos campos.

Este formulario está potenciado con Bootstrap, y la longitud de los campos de texto están adaptados según a lo que se espera introducir. Es decir, mientras que el campo categoría supondrá un tamaño de unas tres palabras, el campo descripción se adapta a siete líneas de área de texto.

Adicionalmente, la modificación de la imagen se presenta en la carta mediante un botón “Cambiar portada”, colocado justo debajo de la imagen por defecto, o la imagen introducida en el formulario de creación de libro.

Así mismo, el escritor podrá observar cómo va quedando la apariencia de su libro, pues mientras realiza los cambios dentro de Detalles, se van actualizando en la carta a la izquierda.

Cuando pulse el botón guardar, se producirá una petición Http Put al controlador, que procesará los cambios en base de datos.

Acceso a listado de capítulos

Los capítulos están organizados de manera que cada uno de ellos tenga presencia propia en un listado, de esta forma se consigue que el escritor localice el contenido con mayor facilidad al tenerlo clasificados unitariamente.

Esta información se presenta una vez se selecciona un libro creado y se accede a su pantalla de detalles.

Como ya se menciona anteriormente, la disposición de esta pantalla reserva un espacio con un tabulado a la derecha, en la cual aparece una cabecera denominada “Capítulos”.

Esta no es la cargada por defecto al seleccionar un libro, pero solo basta con seleccionar la cabecera para visualizarla. Es entonces cuando se accede al listado.

La peculiaridad aquí reside en que el lado cliente no necesita hacer ninguna petición get para obtener los capítulos del libro, pues se hace uso de un almacenamiento local del libro, y como ya se ha explicado anteriormente, el modelo de libro contiene los capítulos.

Basta con acceder al objeto libro, y obtener su propiedad capítulos.

Entonces pueden ocurrir dos cosas:

1- No hay capítulos creados.

En este caso se muestra un mensaje claro y en negrita que incentiva al escritor a crear su primer capítulo. El mensaje figura como:

Aún no tienes ningún capítulo creado.

¿A qué esperas?

2- Hay algún capítulo creado

Para esta situación se genera una lista con tantos elementos como capítulos haya. En cada elemento de la lista aparecerá un título arriba y en grande, una fecha de actualización, y un número de palabras.

Este número se calcula haciendo una lectura del texto del capítulo, y consultando por su longitud.

Crear y borrar capítulos

El desarrollo de esta funcionalidad recicla el enfoque, estilo y diseño de “crear, borrar y editar libros”, para obtener agilidad y mantener una misma línea y procedimiento mental al usuario de la aplicación.

Visualizando la pestaña Capítulos del tabulado, en la pantalla de detalles de libro se tiene:

1- <u>Crear capítulo</u>

Arriba a la derecha se sitúa un botón Nuevo capítulo. Al pulsar el botón se produce una redirección a la pantalla de escritura en línea, y el escritor podrá visualizar el editor de texto vacío, con el nombre del capítulo generado por defecto (Capítulo + Núm. de capítulo).

Esta interacción se diseña así con el fin de ahorrar un paso al escritor, pues no tiene sentido diseñar una opción que permita introducir el título del capítulo o el cuerpo, teniendo ya de por sí el editor que está preparado específicamente para ello.

Paralelamente a lo que ocurre en pantalla, se actualiza localmente la propiedad de capítulos del libro, añadiendo un nuevo capítulo vacío. Con el objeto libro modificado, se envía una petición Http put al controlador de los libros.

Este procede a localizar el libro a modificar, y una vez hecho, lo actualiza con la información del nuevo.

Nótese que no se realiza ninguna petición post, puesto que un capítulo no tiene colección en CosmosDB de por sí, sino que forma parte de la colección Books.

2- Borrar capítulo

Esta funcionalidad es prácticamente idéntica a la de borrar libro. El botón de borrado se hace visible en la lista, y cuando se selecciona, se levanta un modal de confirmación.

El escritor procede con el borrado si pulsa “Ok”, el cual está tintado en rojo para resaltar el riesgo que presenta la acción.

De nuevo, no se procede una petición Http *delete*, puesto que no se está eliminando un libro de la colección, sino que se está actualizando un libro cuya lista de capítulos tiene uno menos.

Por tanto, se procede al borrado en la lista local, se actualiza la referencia y se envía el libro modificado con Http put.

Acceso a escritura en línea

La escritura en línea es una pantalla que permite edición de capítulo, seguimiento de un índice del capítulo, y corrección ortográfica de texto. Se presenta como una pantalla clara, limpia e intuitiva para el escritor y para ello se optan por los colores planos y un buen uso del espacio.

Recalcar que el editor de texto es el elemento principal y protagonista de esta pantalla y por eso toma el centro de la misma.

El acceso a esta pantalla queda reducido a cuando el escritor pinche sobre un capítulo que quiera seguir escribiendo, o como se menciona anteriormente, a cuando se crea un capítulo.

En el momento en el que se accede se cargan los elementos de manera que el guion rápido y el análisis de texto quedan escondidos arriba a la izquierda en un menú desplegable.

Al momento de acceder se carga el componente Quill, cuya integración en Angular se basa en la instalación de dos paquetes mediante el gestor de paquetes de Node.js

npm install ngx-quill

npm install quill

Una vez instalado se tiene acceso a su modulo, que no es más que un api con métodos de control y gestión de texto, y opciones de configuración para los temas que brinda en su apariencia.

Destacar que Quill proporciona dos tipos de temas básicos:

1- Bubble: Simple, sin ningún tipo de encabezado y solo con el cuerpo del editor, cuyas opciones de formato se muestran al seleccionar el texto escrito con el ratón.

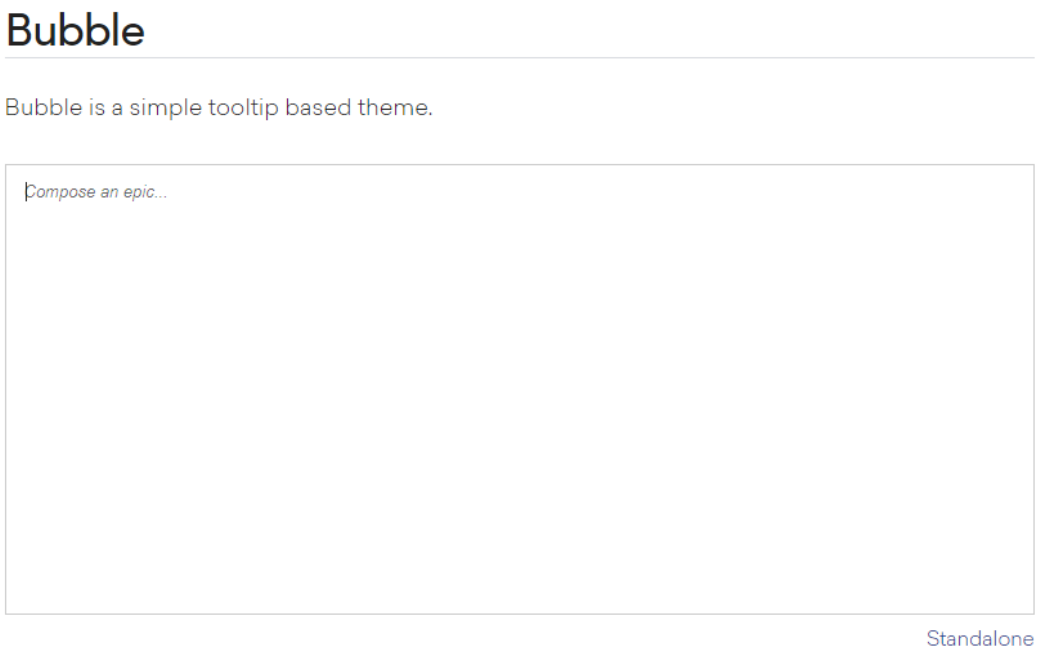


Figura 21 - Ejemplo editor Bubble.

2- Snow: Limpio, con una cabecera de opciones básicas de estilo que se aplican sobre el texto escrito. Al seleccionar el texto no se muestran opciones, y se aplican solo desde la cabecera.

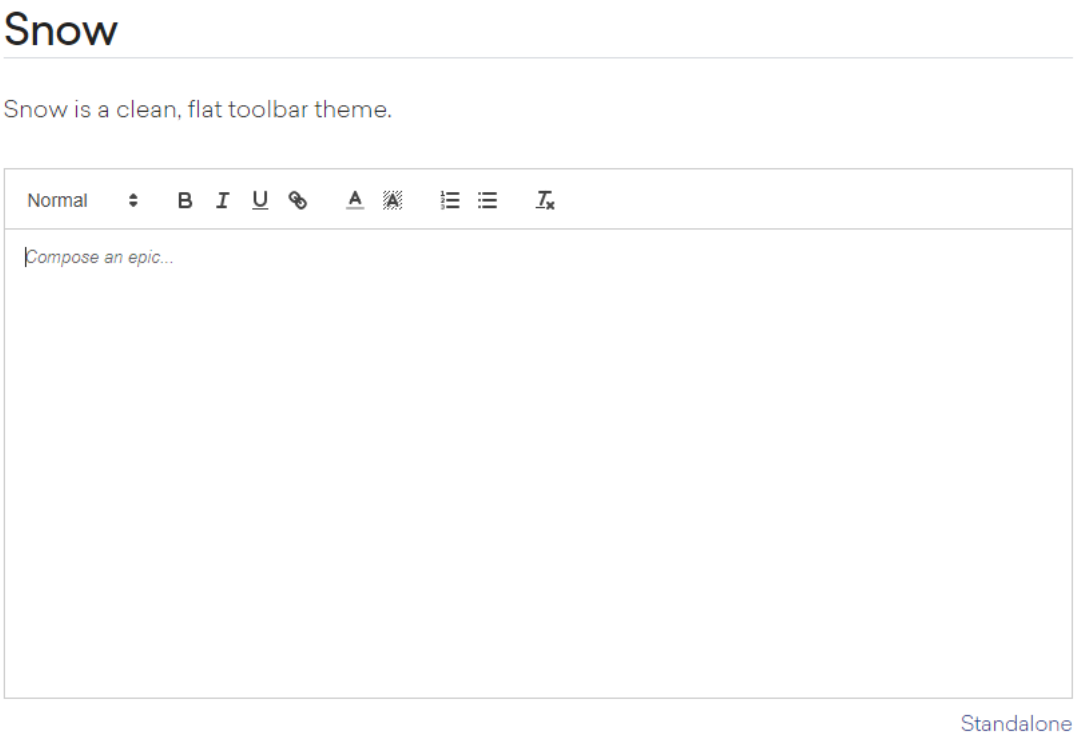


Figura 22 - Ejemplo editor Snow.

Adicionalmente, se puede personalizar una apariencia propia, y es esta la opción que se elige en el proyecto.

De cualquier forma, Quill ofrece una apariencia atractiva y minimalista.

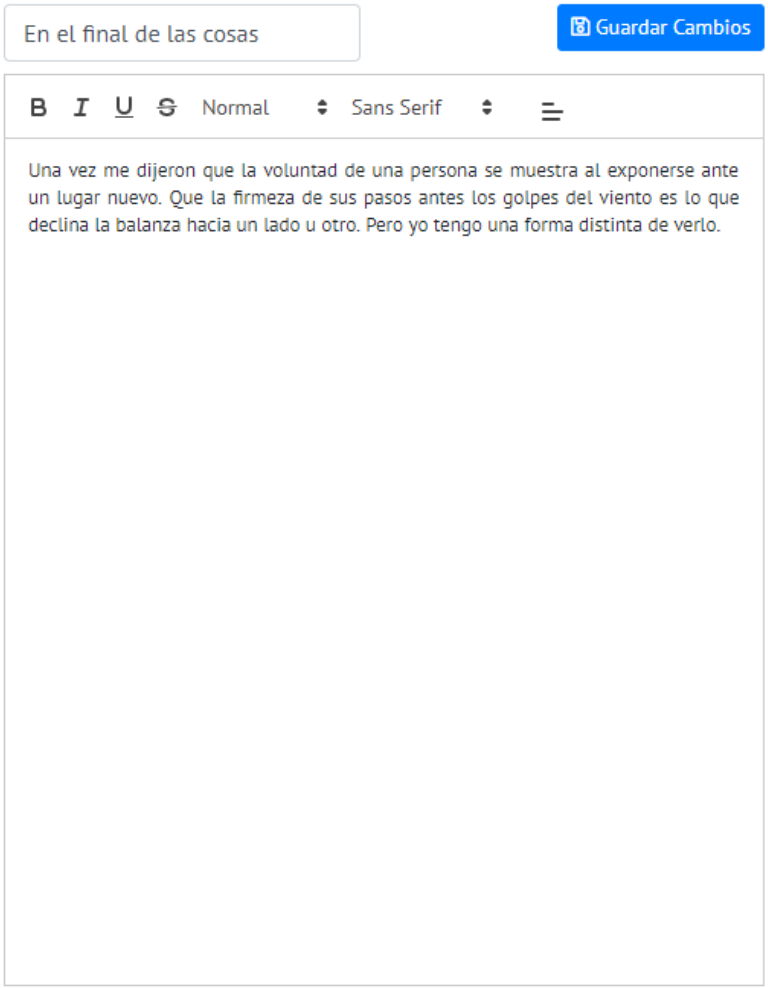


Figura 23 - Editor de texto Quill final

Dentro del espacio de texto se incluye el contenido del capítulo, y justo por encima del editor aparece una caja de texto con el título del capítulo y una opción de guardado a la derecha.

Editar capítulos en un editor de texto

Quill presenta numerosos métodos que capturan eventos de la escritura.

- (onEditorCreated): Lanzado en cuanto el componente se carga en pantalla. Se puede usar para capturar el texto del capítulo.
- (onContentChanged): Lanzado en cuanto el texto del editor es modificado. Supone un gran control de guardado o de contador de palabras.
- (onSelectionChanged): Lanzado cuando se selecciona el editor. Su uso es más puntual, pero se puede sacar partido en situaciones en las que el lugar donde pincha el escritor es importante. Por ejemplo, si se desea mover el cursor del texto hacia donde se pincha.

No obstante, el uso de estos eventos queda reducido al combinarlo con la potencia de Angular. NgModel es una directiva que proporciona un mayor control de formularios, creando una comunicación entre la vista y el modelo. Al hacer uso de ella se puede obtener valor para un campo de entrada desde el modelo, así como el propio modelo obtener el valor modificado de la vista si es que la comunicación se especifica como bidireccional.

"editorContent" es la propiedad que se designa al contenido del editor Quill. Usando el evento de angular ngOnInit (lanzado nada más entrar en la pestaña) se le asigna al contenido del editor, el texto guardado del capítulo.

Con esta mecánica ya no es necesario valerse del primer método de Quill onEditorCreated.

¿Cómo se produce entonces el guardado del capítulo una vez se cambia?

En este caso se combinan dos vías.

1- Guardado automático

Cada vez que el escritor introduce un nuevo carácter en el capítulo, se hace uso del control de evento onContentChanged. Este almacena siempre en local el número de palabras, y el contenido del capítulo, tanto en texto plano como en texto enriquecido.

Ahora bien, si el número de caracteres introducidos excede los 30, se produce el guardado en base de datos, realizando una petición Http put del libro con el capítulo modificado en local.

Para señalizar que se produce el guardado, el botón *"Guardar capítulo"* se pone en inactivo y se produce una animación de carga dentro.

2- Guardado manual

El almacenamiento local se produce mediante el mismo evento mencionado en el guardado automático, solo que, en esta ocasión, al pulsar directamente sobre el botón “Guardar capítulo” situado arriba a la derecha del editor, se realiza la petición Http y el guardado en base de datos.

Remarcar que la facilidad que aporta quill a la hora de obtener el texto plano y el texto enriquecido facilita futuras funcionalidades como la de exportación a formato Word del libro.

Elegir formato de texto y apariencia.

Ya se ha hecho mención de los dos temas que proporciona Quill, y que adicionalmente se puede crear un tema personalizado.

Quill proporciona una gran cantidad de elementos que incluir en la cabecera, desde alineación o poner una palabra en negrita, a tener enumerados o incluir formato para funciones matemáticas o incluso seleccionar la dirección de escritura de texto.

Tan solo con incluir la etiqueta <quill-editor></quill-editor> en la vista, se carga el editor quill por defecto.

El trabajo reside en adaptarlo para generar una experiencia óptima, y cumplir con los requisitos básicos de edición.

En primer lugar, se adapta el tamaño para que aparezca de manera perpendicular al igual de lo que podría ser una hoja Word.

En el propio etiquetado de quill, es donde entra la personalización de la cabecera, añadiendo los atributos “class” correspondientes al etiquetado *quill-editor-toolbar.* Una vez se selecciona cualquiera de los atributos incluidos, se asigna su valor, y Quill lo reconocerá y aplicará al cuerpo del editor.

Los atributos asignados según el requisito funcional son los siguientes:

1- Para el tamaño

Se asigna el atributo “ql-size” con los valores 'small', 'large' y 'huge', por defecto se deja un valor Normal, intermedio entre small y large.

2- Para la alineación

Se asigna “ql-align” con los valores ‘center’, ‘right’, ‘justify’, por defecto se deja el texto a la izquierda.

3- Negrita, cursiva, subrayado

Cada opción tiene un atributo propio, “ql-bold”, “ql-italic”, “ql-underline”, “ql-strike”, para negrita, cursiva, subrayado y, adicionalmente, tachado. Por defecto no se selecciona ninguna de ellas.

4- Tipografía

En este caso había numerosas opciones que añadir. “ql-font” presenta por defecto las tipografias serif y sans-serif, por lo que, por simplicidad y tiempo, se decide dejar dichas opciones.

No obstante, en caso de querer añadir alguna tipografía en específico, se debería descargar e incluir en el proyecto previamente y después configurar el atributo desde css.

Exportar libro en formato Word

Esta funcionalidad se hace visible en la pantalla de detalles del libro. En la tarjeta con la información general y portada, aparece un botón “Exportar libro” bien resaltado en naranja.

Cuando el usuario pulsa dicho botón visualiza una descarga, y paralelamente a esta interacción, se realiza todo el procesado de los capítulos del libro.

Lo primero que hay que tener en cuenta en este procesamiento de texto es que no se debe trabajar con el texto enriquecido que se obtiene del editor, sino que hay que hacerlo con el texto plano. Esto se realiza con el fin de no trasladar etiquetado propio de la página html al documento Word final.

Debido a problemas de comunicación entre la construcción del documento, y la descarga del navegador, no hubo otra opción que hacer uso de un componente externo que centralizase todo el trabajo en el lado cliente.

Para proceder con ello, se hace uso de un componente externo denominado “docx”, cuya integración consiste en una instalación mediante el gestor de paquetes npm, y la correspondiente importación de clases en el módulo en el que se pretende usar.

Destacar la clase *Document* para construir el documento, y la clase Packer para su descarga en el navegador.

Document permite realizar una escritura en un documento Word, controlando aspectos cómo creación de párrafos, estilos, tamaños de texto, tipografía o espaciado entre otras.

Las funciones de creación de párrafos toman el texto y aplican un estilo. El procesado de texto que se decide tomar es simple, y eficaz.

En primer lugar, se crea un párrafo para el título y se le aplica un estilo personalizado con un espaciado y tamaño diferenciador.

Se prosigue con la lectura del contenido del capítulo, y para ello, se crea un simple algoritmo de iteración que generará un párrafo cada vez que encuentre la cadena “\n” en el texto del capítulo.

Una vez creado el documento se usa la clase Packer que generará un archivo .doc con el nombre del libro.

En caso de que el libro no tenga capítulos aparecerá un documento vacío.

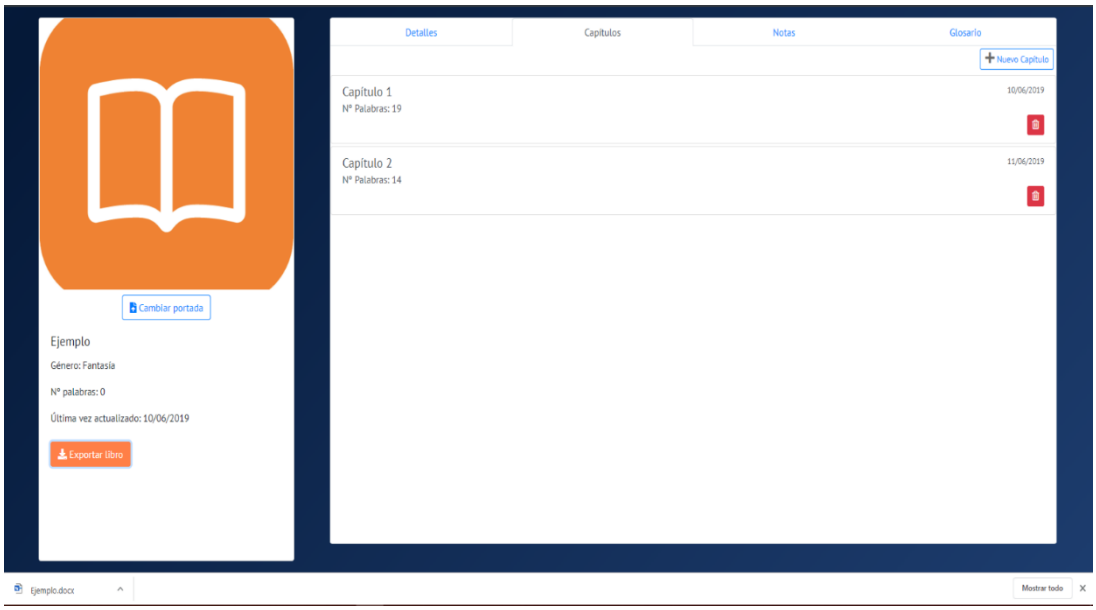


Figura 24 - Ejemplo de exportación libro, descarga.

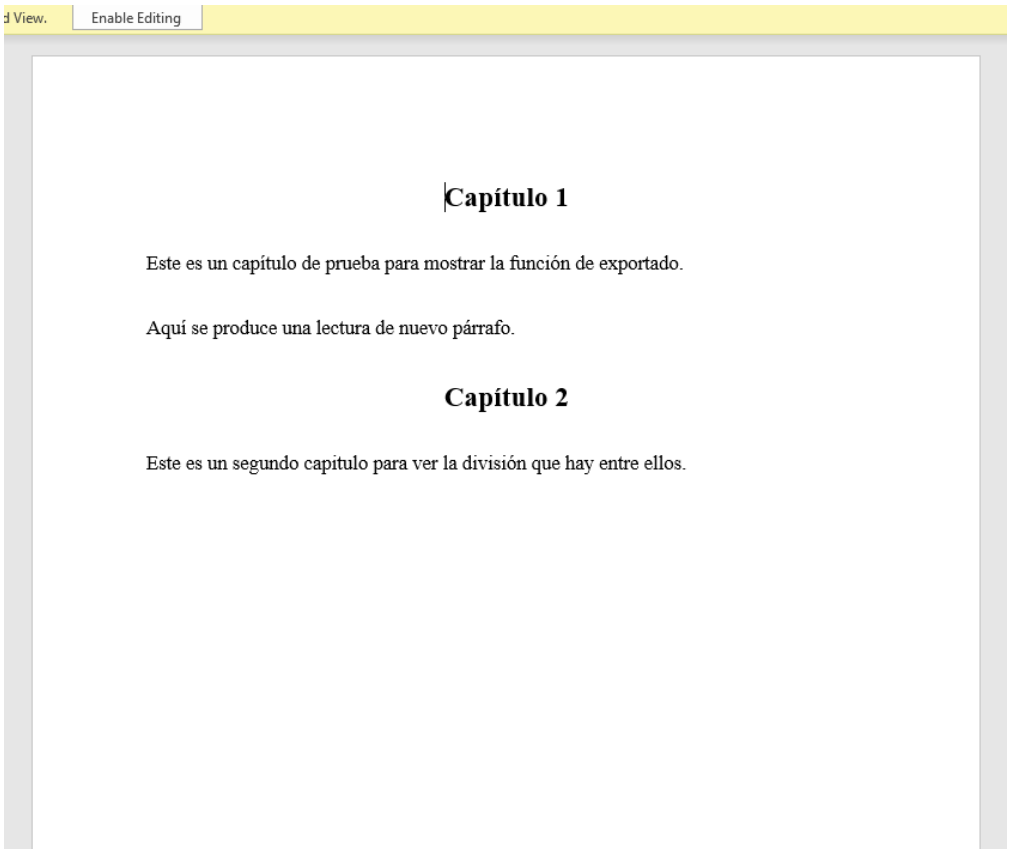


Figura 25 - Ejemplo de documento exportado

Corrección de ortografía

Este requerimiento nace de la idea de retroalimentar la escritura del escritor a nivel ortográfico, pero lo que en un principio se creía como una tarea de enfoque claro, durante el desarrollo se encontraron dos sorpresas que supusieron un **replanteamiento y trabajo de adaptación**.

1- El impreciso resultado de incorporar BingSpellCheck:

   Por su vasto contenido y la agradable experiencia de uso con CosmosDB, se decidió trabajar con los servicios de Azure, y usar sus APIs cognitivas para el procesamiento y análisis de texto. No obstante, a pesar de la potencia que se podía alcanzar, a pesar de el gran abanico de posibilidades que Azure proporciona, el uso de sus APIs para este cometido acabó siendo pobre.

   Tras realizar la integración y comunicación con la API BingSpellCheck, para detectar las faltas de ortografía, aparecieron numerosas imprecisiones a la hora de detectar las reglas del lenguaje español.

   Tildes irreconocibles, sugerencias de corrección ilógicas… Quedó claro, que se necesitaba ir un paso más allá y el planteamiento de otras alternativas comenzó poco después.

2- El navegador proporciona corrección ortográfica efectiva si se activa en su configuración:

   Descubrir este hecho supuso un duro golpe funcional al requisito, puesto que con solo un activar una opción, se puede reconocer las faltas a tiempo real que se producen mientras se escribe. Esta opción está disponible para la gran mayoría de navegadores actuales.

Tras estos dos hechos, ¿de qué manera se iba a proporcionar una solución que tuviera sentido funcional?

Cuando se descubre Stilus, una API concretamente dedicada al análisis de texto que proporciona detección de faltas de ortografía, gramática, reglas de puntuación o incluso el estilo, la funcionalidad vuelve a cobrar fuerza.

Esto se debe al tipo de respuesta que Stilus devuelve ante una petición. Desde el texto con el error incluido o las sugerencias de corrección que ya se tenían en otras Apis, hasta el tipo de error lingüístico, el nivel de habla en la que se sitúa el error, o el mensaje explicativo del error que hace esta Api exclusiva.

Este tipo de información potencia el aspecto retroalimentativo hacia el escritor, y marca un punto diferenciador con lo que la corrección del navegador puede aportar.

Además, refuerza el conjunto de correcciones posibles, pues si el diccionario de sugerencias del navegador no reconoce una falta, Stilus puede proporcionarla, y viceversa. Por ejemplo, la palabra amarronado es detectada como error por el navegador cuando Stilus sí que la tiene incluida en su diccionario de palabras correctas. Entre las sugerencias que proporciona el corrector en línea, tampoco está dicha palabra. Véase en la imagen:

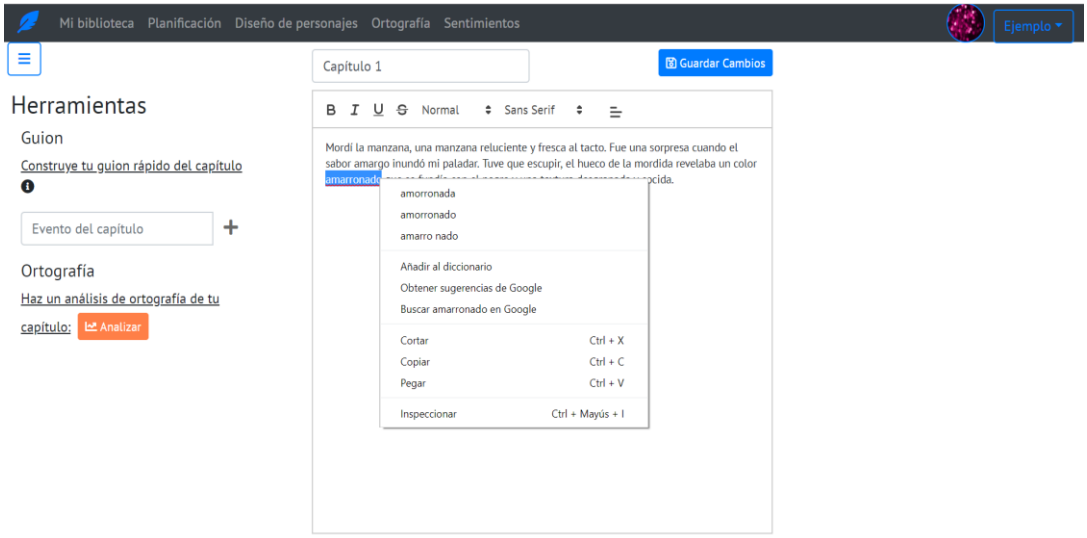


Figura 26 – Evidencia corrección navegador

Chrome, en este caso, detecta un error inexistente en amarronado, pero si se procede a analizar las faltas con Stilus:

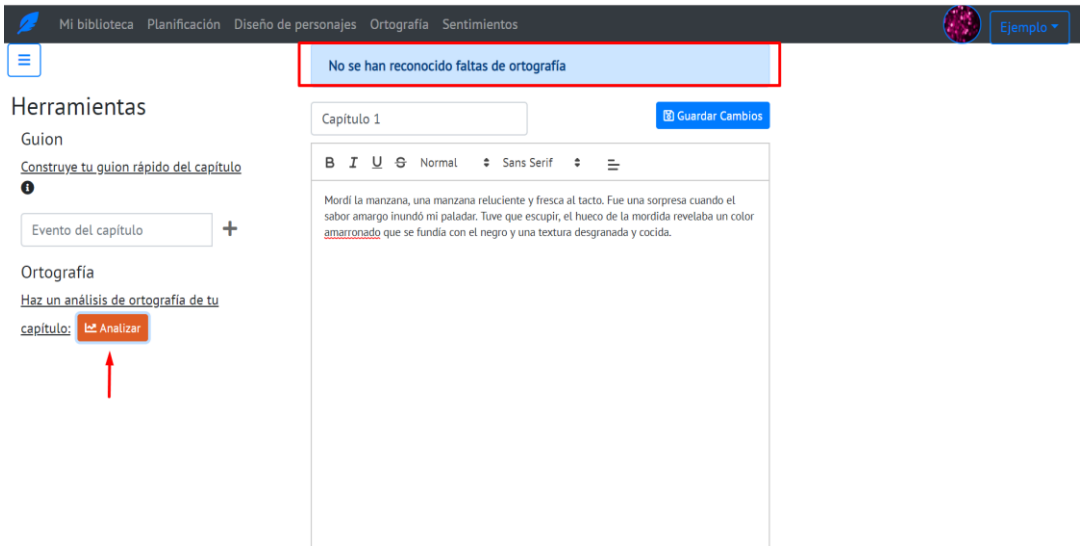


Figura 27 – Evidencia corrección por Stilus

Esto genera más consistencia.

Pero este motivo no fue el único por el que se siguió apostado por la API de corrección de texto. La combinación con gráficas de análisis se convirtió en una idea realmente atractiva y relativamente fácil de abarcar.

Los navegadores muestran las faltas a tiempo real, pero su almacenamiento no es aparentemente visible, y obtener análisis a partir de ello se vuelve una tarea mucho más compleja de abarcar que usando una API como Stilus y utilizar la respuesta para aplicarla en una gráfica.

La decisión pues, fue clara: aplicar Stilus por su retroalimentación al escritor, además de activar por defecto la opción de corrección de los navegadores con una propiedad de Html.

Así se completaría la corrección por dos vías.

A nivel de **implementación y diseño**, esta funcionalidad se hace visible tanto en la escritura en línea como en el apartado de análisis (se profundiza en el siguiente punto). Consiste en incluir en ambas pantallas un botón resaltado en naranja con el texto “Analizar ortografía”, que al ser pulsado establece la comunicación con la Api y devuelve una lista de faltas detectadas, mensaje explicativo, y sugerencias de corrección. En cada elemento de la lista aparece un botón de corregir que sustituirá la sugerencia en el texto original.

Esto último se acompaña con un mensaje de completado tras ello.

Indicar, que el diseño se respeta en ambas pantallas, con la única salvedad de que, en la escritura en línea, las faltas de ortografía no se almacenan en base de datos y se respeta la misma lógica que el corrector del navegador toma.

Esto se decide por el alto carácter dinámico de la escritura en línea, en el sentido de que siempre se está modificando el texto; y también porque la principal utilidad de las faltas de ortografía es informar al momento de lo que ocurre. De esta forma, se recibe la información por pantalla, se corrige el texto al momento con dicha información y se actualiza.

Para este proceso no es necesario acceder a base de datos, hacerlo sería inútil y poco eficiente, se cargaría el sistema con numerosas peticiones.

El resultado puede visualizarse en las siguientes imágenes:

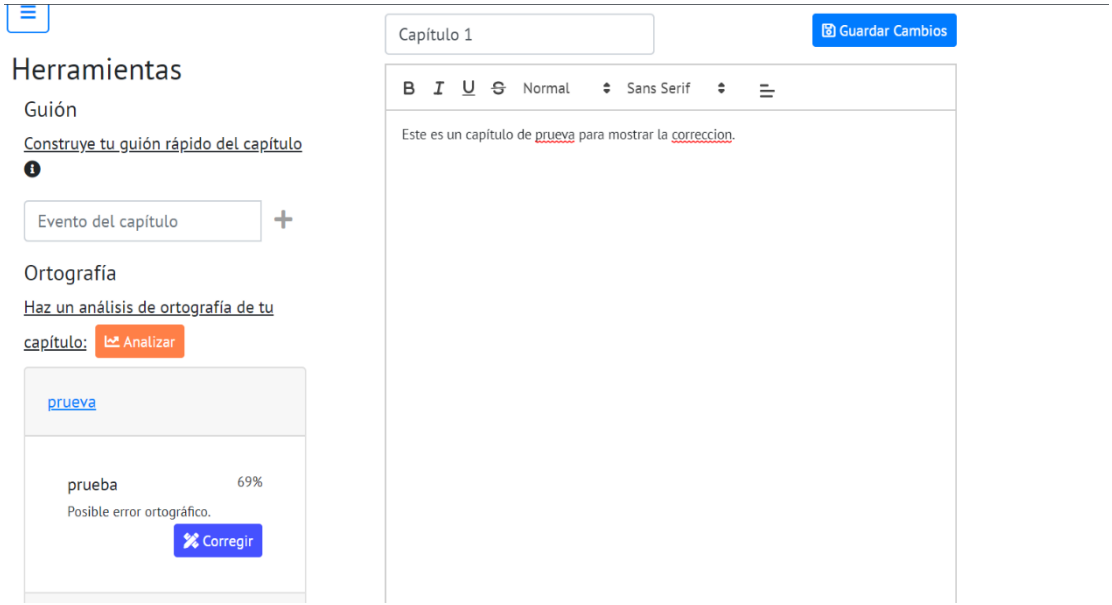


Figura 28 – Detección de erratas en el editor

Tras pulsar “Corregir”:

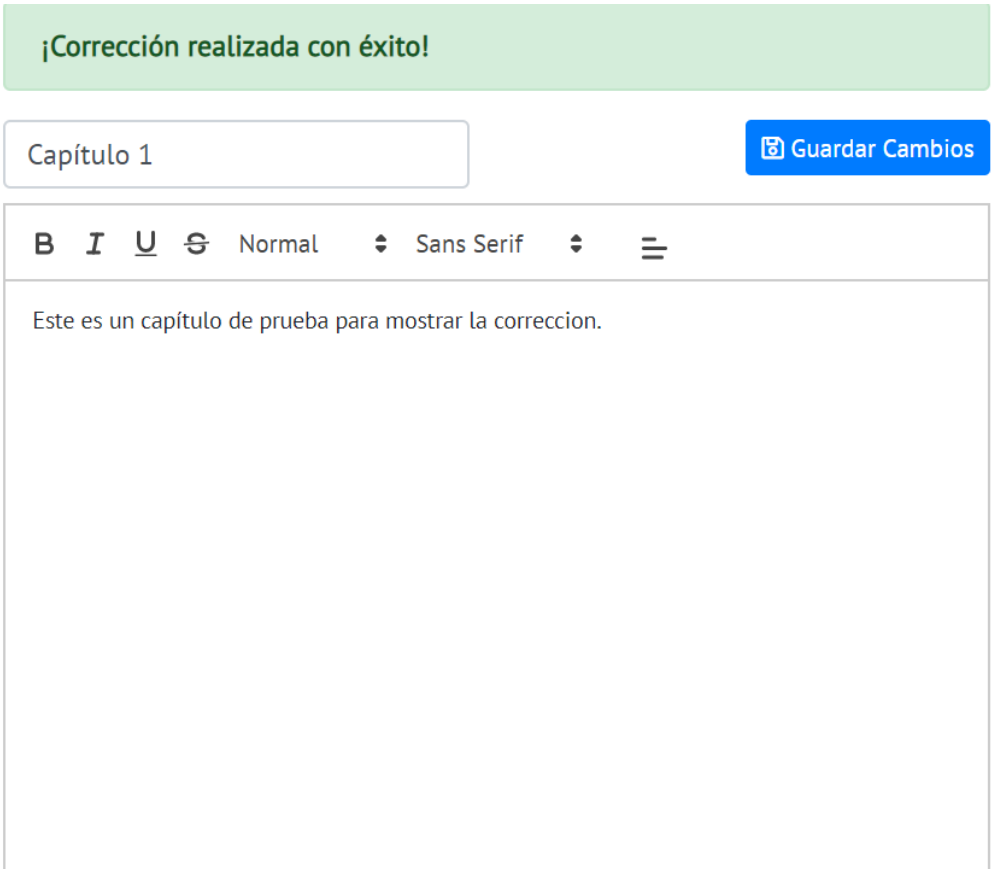


Figura 29 – Corrección realizada con éxito

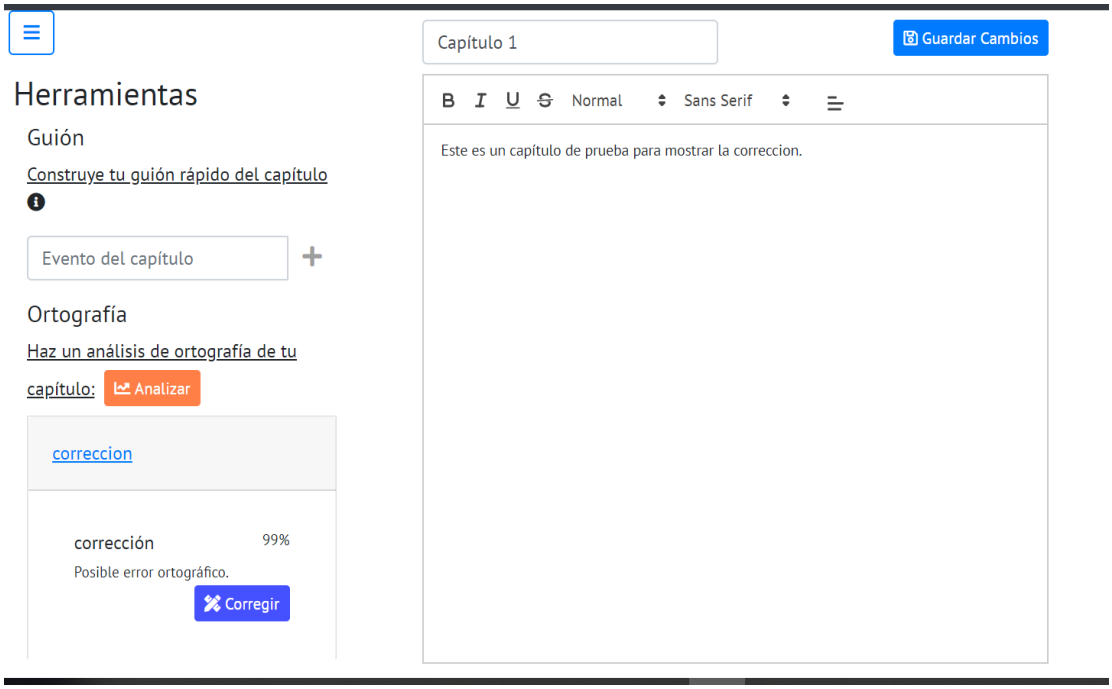


Figura 30 – Estado final tras corrección

A nivel de servidor, la dificultad de esta comunicación con la API de Stilus reside en que la alta cantidad de información que esta devuelve hay que recogerla en modelos, traducirla y enviarla al lado cliente.

Se tuvieron que adaptar cuatro modelos para conseguir recoger toda esta información de forma consistente, cada compuesto del anterior.

StilusSugestion -> StilusResult -> StilusCorrection -> TextAnalytic

Gráfica de análisis de ortografía

La generación de gráficas queda cubierta con la librería Charts.js, que proporciona un conjunto de gráficas flexibles y simples construidas en JavaScript, y de fácil integración con los componentes de Angular que requieren su uso.

Line chart fue la gráfica elegida para la representación de faltas de ortografía. En la documentación oficial fue fácil ver un ejemplo de los parámetros que se debían informar, así como el resultado final esperado.

Se decidió que el análisis se realizase por libros completos, detectando capítulo a capítulo qué faltas de ortografía tenían. Por lo tanto, en el eje *y* de la gráfica se muestra el número de faltas cometidas, y en el eje *x* se muestran los capítulos que componen el libro. Esta gráfica unirá entre sí puntos que se asignan, mostrando así una línea de mejora o empeora.

En cuanto a la interfaz, el escritor contará con el acceso a la pantalla *ortografía.* La composición de dicha pantalla es la siguiente:

- Arriba se muestra un seleccionable que contiene los libros disponibles a analizar.
- En el cuerpo de la pantalla, a la izquierda y en grande, se muestra la gráfica de análisis.
- A la derecha se muestra la información referente del análisis, ya sean capítulos con faltas a corregir, o un mensaje informativo del resultado en caso de que no se hayan detectado faltas o no se haya realizado análisis.

Figura 31 – Pantalla de análisis de ortografía

Nada más el escritor accede a dicha pantalla podrá visualizar una gráfica cargada por defecto, y cuando seleccione un libro a analizar, o bien se carga la información del último análisis si es que el libro se analizó previamente, o, por el contrario, se carga una gráfica base de dicho. Esta, contendrá tantas divisiones en el eje x como capítulos tenga el libro, y el eje *y* estará inicializado a cero.

Figura 32 – Formato de gráfica por libro

Para proceder con un análisis, el escritor deberá pulsar el botón “Analizar”, y con ello, se enviará una petición Http Post con el contenido de los capítulos del libro seleccionado a la API de Stilus, la cual devolverá la información sobre las erratas del escritor.

Esta respuesta es transformada y pasada a un modelo adaptado, de manera que se incluye información útil como el Id del capítulo analizado, del libro y del escritor.

Cuando el controlador de este servicio cognitivo devuelve la respuesta final, el lado cliente hace un recuento de las faltas detectadas por capítulo y transmite esta información a la gráfica.

Como ya se ha mencionado en el punto anterior, en caso de que se hayan detectado erratas, se pueden corregir mediante el mismo mecanismo de corrección usado en la pantalla de escritura en línea.

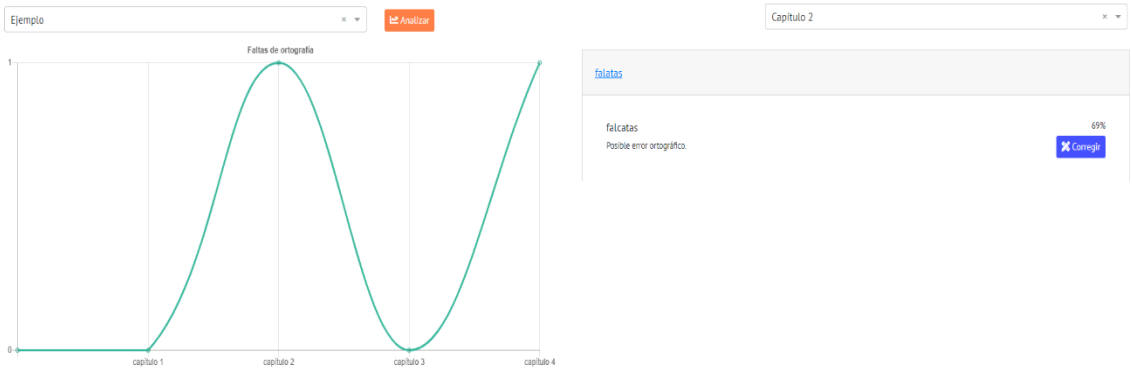


Figura 33 – Gráfica con faltas detectadas

Como añadido, si una falta es corregida, la gráfica se actualizará, haciendo así más visible el proceso de corrección.

Una vez se produce el análisis, este queda almacenado en la base de datos, siendo posible que se rescate en un futuro por si el escritor no termina de realizar su corrección, además de aportar un punto de comparativa del estado en el que quedó su libro con el que se encuentra actualmente.

Guion rápido de los capítulos

Esta funcionalidad se abarca con el fin de hacer que el escritor tenga un esquema mental del capítulo plasmado en la misma pantalla donde escribe.

Se enfoca de manera que no dificulte la visión del editor de texto, y por se incluye dentro de un desplegable de herramientas que por defecto plegado.

Cuando el escritor pulsa sobre el menú, se despliega y muestra la sección de guion, donde el escritor puede introducir en una caja de texto el evento que va a suceder. *Ver figura 9*.

Se incluye, además, un mensaje de ayuda con un ejemplo de evento de guion para evitar posibles confusiones.

Figura 34 – Mensaje informativo guion rápido

El botón de añadir no generará nada a menos que se introduzca un carácter en la caja de texto.

Cuando se incluye un evento en el guion rápido se añade directamente a una lista, en la que cada elemento tiene una caja de confirmación a la izquierda para que pueda ir marcando los eventos que ha desarrollado en el capítulo.

A la derecha aparece un botón de borrado, que esta vez, al ser pulsado, no muestra ventana de confirmación. Esto se decide así, puesto que un evento del guion no es un elemento crítico, y de esta forma se consigue minimizar las interacciones que tiene el escritor con la página y agilizar el proceso.

Con el fin de separar el mecanismo de borrado con lo que se había mostrado anterior mente en la aplicación, se decide cambiar la apariencia del botón a una distinta, representando así un cambio de metáfora.

El botón de borrado con ventana de confirmación se representa con una papelera, mientras que el botón de borrado sin confirmación se representa con una ‘x’.

Figura 35 – Borrar evento incluido en lista

Además, los botones se simplifican al máximo para que no ocupen demasiada pantalla, y mantener el estilo minimalista que se buscaba conseguir.

Con respecto a la comunicación entre cliente y servidor, las peticiones que ocurren paralelamente a esta interacción se presentan de igual forma que en otras funcionalidades descritas anteriormente: Un evento de guion forma parte de un capítulo, que forma parte de un libro, por tanto, la petición Http final es de tipo Put hacia la colección de libros.

Subida y visualización de imágenes

Esta funcionalidad presentó ciertas dificultades a la hora de desarrollarla, y se necesitó un tiempo de investigación para acometerla.

Se plantearon varias opciones: almacenar la imagen en un repositorio externo y hacer referencia, almacenar la imagen directamente en la base de datos, y almacenar la imagen en el servidor y guardando su ruta en base de datos para hacer referencia a ella posteriormente.

Esta última es la más eficiente a nivel de espacio y rendimiento, por lo que fue la opción que se acabó tomando.

Para ello se abrió la posibilidad de incorporar componentes externos en el lado cliente que facilitaran la obtención directa de la imagen y su envío al servidor, pero ninguna de las opciones respondió como se esperaba, y si bien ng2-file-upload apuntaba buenas maneras con un conjunto de eventos de captura realmente útiles para validaciones, acabó consumiendo demasiado tiempo de integración y documentación en un periodo del proyecto que empezaba a ser crítico.

Se decidió por trabajar con Bootstrap para obtener la imagen y hacer trabajo de modelo y comunicación Http con el lado de servidor .Net donde se usaría la clase FormFile para el almacenamiento de la imagen en el servidor wwwroot.

Finalmente se almacenará la ruta en una propiedad del modelo libro.

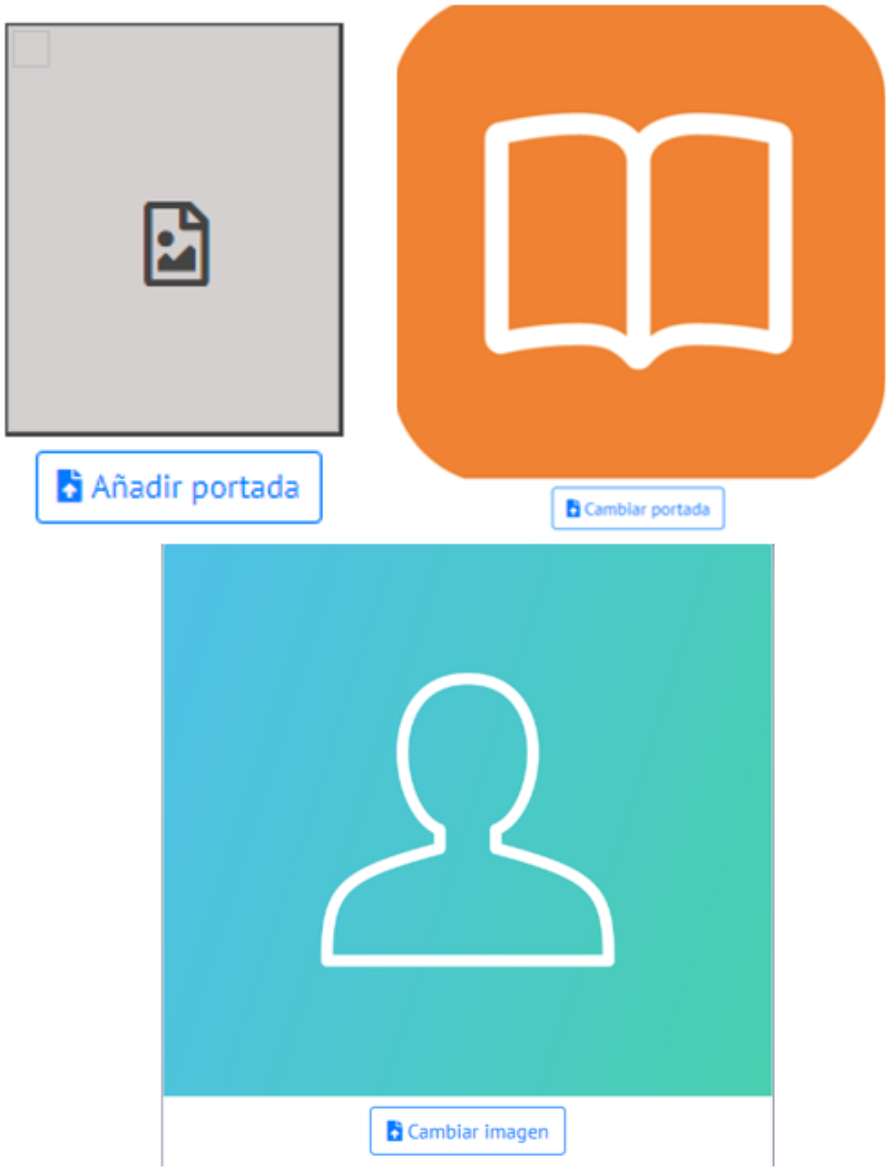


Figura 36 – Subida de imágenes

Esta funcionalidad es reciclada para ser usada en distintas pantallas, presentando la misma interacción con el escritor. Se cumple siempre con la misma mecánica, presentando una imagen por defecto en el espacio reservado hasta que el escritor decida subir una propia.

7

Pruebas

Pruebas unitarias

La realización de pruebas para una aplicación es una fase que hay que tomar muy en consideración, tanto es así, que incluso existen metodologías enfocadas a la realización de pruebas en primer lugar antes de lanzarse a desarrollar. O, dicho de otra forma, realizar un desarrollo basado en un escenario de pruebas.

Esta fase ayuda a conseguir un código de calidad, comprobar el correcto funcionamiento de la aplicación, estudiar el comportamiento del sistema frente a distintas situaciones. De esta forma se logra mayor estabilidad y mantenibilidad, y al mismo tiempo, se verifica si el proyecto ha cumplido con los objetivos previamente establecidos.

Para conseguir este resultado, la fase de pruebas proporciona un gran número de variantes, entre las cuales están los test unitarios, test de estrés, test de integración, test de usabilidad… entre otros. Cada tipo de prueba cubre un aspecto distinto de la aplicación, proporcionando una información lo más objetiva posible en forma de un resultado numérico, a partir del cual, se toman medidas y se buscan soluciones en caso de que se necesiten.

Para esta etapa de pruebas se ha enfocar el esfuerzo sobre las pruebas unitarias de la Api, que es la encargada de conectarse a la base de datos y enviar los datos correspondientes a la parte del cliente. Se ha acordado realizar solo estás pruebas porque eran las que se ajustaban mejor al tiempo que tenía el proyecto planificado.

La guía principal que se ha seguido para la realización de cada una de las pruebas ha sido el principio de las tres A’s. Este principio hace que cada test se divida a su vez en tres partes diferenciadas:

- Arrange (Organización): Esta etapa sirve para hacer toda la preparación de los elementos del sistema que van a intervenir y ser probados en la prueba a realizar.
- Act (Actuación): Etapa donde se realizan las llamadas a los métodos correspondientes.
- Assert (Comprobación): Etapa donde se verifican y comprueban que los resultados a las llamadas de la etapa anterior son correctos.

Estás pruebas unitarias han sido de gran utilidad a la hora de mantener la Api con un código más limpio y de calidad, eliminando duplicidad de código y métodos sin utilidad, haciendo que dichos controladores sean más manejables,

escalables y más sencillos de mantener. Para cada acción de los controladores que se usa en la Api, Post, Put, Get y Delete, se han realizado pruebas unitarias cubriendo todos los caminos que podía tomar el flujo de control.

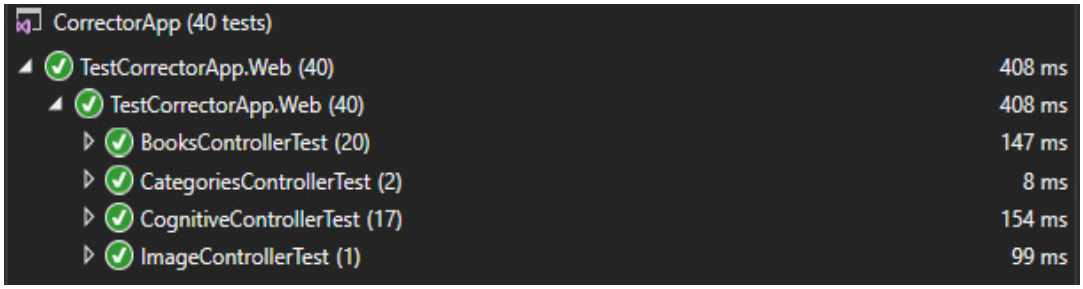


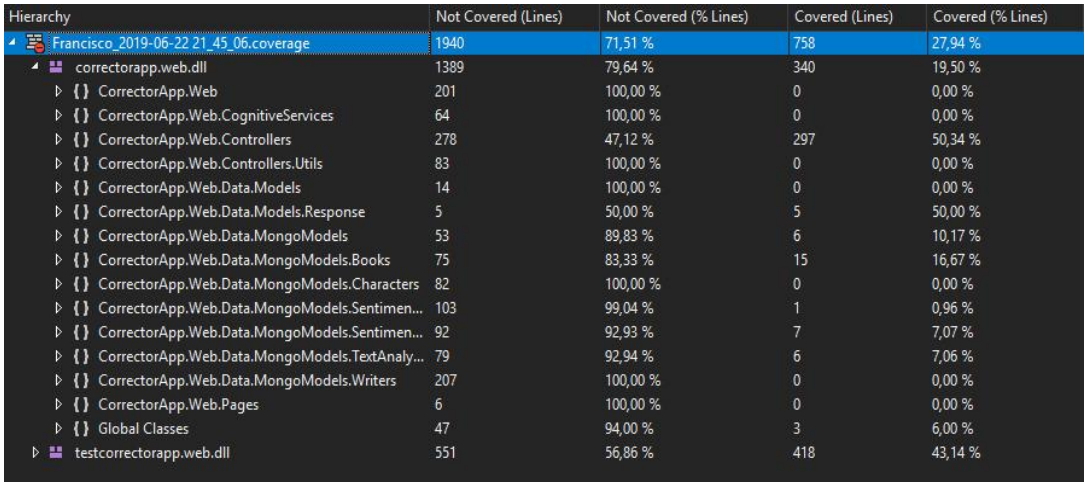
Figura 37 – Pruebas unitarios

Se observa la construcción de cuarenta pruebas unitarias, que hacen que los controladores involucrados en la parte del proyecto que aquí se desarrolla, tenga una calidad y seguridad de que cumple con los requisitos establecidos en los objetivos. Durante la realización de estas pruebas se puso especial atención en marcar los límites de lo que se debía probar y lo que no. Los servicios de terceros se dan por sentado que funcionan correctamente, pues no es la responsabilidad de este proyecto verificar dicha funcionalidad. De esta forma se evita el enfrentamiento con las cajas negras.

Hierarchy	Not Covered (Blocks)	Not Covered (% Blocks)	Covered (Blocks)	Covered (% Blocks)
Francisco_2019-06-22 21_45_06.coverage	3057	71,09 %	1243	28,91 %
correctorapp.web.dll	1859	81,97 %	409	18,03 %
CorrectorApp.Web	179	100,00 %	0	0,00 %
CorrectorApp.Web.CognitiveServices	83	100,00 %	0	0,00 %
CorrectorApp.Web.Controllers	354	49,17 %	366	50,83 %
CorrectorApp.Web.Controllers.Utils	81	100,00 %	0	0,00 %
CorrectorApp.Web.Data.Models	14	100,00 %	0	0,00 %
CorrectorApp.Web.Data.Models.Response	5	50,00 %	5	50,00 %
CorrectorApp.Web.Data.MongoModels	51	89,47 %	6	10,53 %
CorrectorApp.Web.Data.MongoModels.Books	126	89,36 %	15	10,64 %
CorrectorApp.Web.Data.MongoModels.Characters	82	100,00 %	0	0,00 %
CorrectorApp.Web.Data.MongoModels.Sentimen...	103	99,04 %	1	0,96 %
CorrectorApp.Web.Data.MongoModels.Sentimen...	175	96,15 %	7	3,85 %
CorrectorApp.Web.Data.MongoModels.TextAnaly...	141	95,92 %	6	4,08 %
CorrectorApp.Web.Data.MongoModels.Writers	404	100,00 %	0	0,00 %
CorrectorApp.Web.Pages	14	100,00 %	0	0,00 %
Global Classes	47	94,00 %	3	6,00 %
testcorrectorapp.web.dll	1198	58,96 %	834	41,04 %

Figura 38 – Cobertura de bloques

Todas estas métricas de cobertura de líneas y bloques están basadas en las pruebas unitarias a cada controlador, y por lo tanto dan una información muy valiosa y visual a simple vista donde se pude ver en que partes del proyecto se aplican las pruebas y que componentes son las que validan dichas pruebas.



Hierarchy	Not Covered (Lines)	Not Covered (% Lines)	Covered (Lines)	Covered (% Lines)
Francisco_2019-06-22 21_45_06.coverage	1940	71,51 %	758	27,94 %
correctorapp.web.dll	1389	79,64 %	340	19,50 %
CorrectorApp.Web	201	100,00 %	0	0,00 %
CorrectorApp.Web.CognitiveServices	64	100,00 %	0	0,00 %
CorrectorApp.Web.Controllers	278	47,12 %	297	50,34 %
CorrectorApp.Web.Controllers.Utils	83	100,00 %	0	0,00 %
CorrectorApp.Web.Data.Models	14	100,00 %	0	0,00 %
CorrectorApp.Web.Data.Models.Response	5	50,00 %	5	50,00 %
CorrectorApp.Web.Data.MongoModels	53	89,83 %	6	10,17 %
CorrectorApp.Web.Data.MongoModels.Books	75	83,33 %	15	16,67 %
CorrectorApp.Web.Data.MongoModels.Characters	82	100,00 %	0	0,00 %
CorrectorApp.Web.Data.MongoModels.Sentimen...	103	99,04 %	1	0,96 %
CorrectorApp.Web.Data.MongoModels.Sentimen...	92	92,93 %	7	7,07 %
CorrectorApp.Web.Data.MongoModels.TextAnaly...	79	92,94 %	6	7,06 %
CorrectorApp.Web.Data.MongoModels.Writers	207	100,00 %	0	0,00 %
CorrectorApp.Web.Pages	6	100,00 %	0	0,00 %
Global Classes	47	94,00 %	3	6,00 %
testcorrectorapp.web.dll	551	56,86 %	418	43,14 %

Figura 39 – Cobertura de líneas

Además, Visual Studio 2017 ofrece algunas herramientas integradas que hacen este proceso muy cómodo y sencillo, presentando de forma muy visual los bloques/líneas de código por donde los test unitarios han ejecutados las pruebas, en azul, y qué líneas de código aún no han sido cubiertas por ningún test, en rojo.

```
// POST api/books/newBook - creates a new book
[HttpPost("newBook")]
4 references | 4/4 passing | francisco.j.rando, 6 days ago | 2 authors, 11 changes | 0 requests | 0 exceptions
public IActionResult Post([FromBody]Book newBook)
{
    try
    {
        if (string.IsNullOrEmpty(newBook.Description))
        {
            newBook.Description = "[Sin descripción]";
        }
        if (string.IsNullOrEmpty(newBook.Title))
        {
            newBook.Title = "[Sin título]";
        }
        if (string.IsNullOrEmpty(newBook.Genrer))
        {
            newBook.Genrer = "[Sin género]";
        }
        newBook.BookId = GetCountDocuments(newBook.WriterId).Result.ToString();
        newBook.UpdateOn = DateTime.Now.ToString("dd/MM/yyyy");
        _bookRepository.AddBook(newBook);
        return Ok(new HttpResponse<Book>()
        {
            Code = 201,
            Status = StatusHttpResponse.Created,
            IsFaulted = false,
            Message = $"Libro {newBook.Title} creado correctamente.",
            ListObjects = new List<Book> { newBook }
        });
    }
    catch (Exception ex)
    {
        return NotFound(ex.Message);
    }

}
```

Figura 40 – Cobertura de líneas en controlador

Para la realización de estas pruebas unitarias se ha utilizado un paquete Nuget llamado Xunit. Este paquete, y su integración en Visual Studio 2017, permite desarrollar pruebas unitarias a una gran velocidad, verificando y encontrando errores en el código que de otra forma sería imposible detectar.

Prueba de usabilidad

Con el fin de obtener una retroalimentación de la experiencia de usuario con respecto a esta web, se realiza un test de usabilidad para tres tipos de usuarios: inexperto, medio, experto. Se ha tomado referencia del sistema de escala de usabilidad (System usability scale) por ser un estándar simple y exacto de diez preguntas con un sistema de respuestas comprendidas del 0 al 5, donde 0 es “completamente en desacuerdo” y 5 es “completamente de acuerdo”. A continuación, se muestra la tabla que recoge las preguntas y correspondientes respuestas de cada usuario.

Preguntas	Usuario inexperto	Usuario medio	Usuario experto
Creo que me gustaría utilizar este sistema frecuentemente.	3	5	4
El sistema me resultó innecesariamente complejo.	3	3	2
Creo que el sistema es bastante fácil de utilizar.	2	3	4
Creo que necesitaría el soporte de un técnico para poder utilizar este sistema.	5	5	1
Creo que las diferentes funciones del sistema se encuentran muy bien integradas.	4	5	4
Opino que hubo demasiada inconsistencia en el sistema.	1	1	1
Imagino que la mayoría de las personas aprendería a utilizar el sistema rápidamente.	2	4	5
Me sentí algo incómodo al utilizar este sistema.	2	1	1
Me sentí muy seguro al utilizar este sistema.	5	5	5
Necesito aprender muchas otras cosas antes de poder utilizar correctamente el sistema.	5	5	4

Tabla de prueba de usabilidad

8

Posibles extensiones

El proceso creativo de escribir un libro genera un gran espacio para admitir vías de desarrollo de la historia, esto acaba traduciéndose en extensiones y nuevas funcionalidades de herramientas.

Con tan solo realizar algunas tormentas de ideas, se lograron recopilar múltiples mejoras:

1- Aviso por correo electrónico de las tareas planificadas.
2- Incluir buscador de palabras, sinónimos y antónimos y poder añadir la palabra directamente al diccionario personal.
3- Poder seleccionar una palabra desde el editor de texto y añadirla al diccionario personal o al glosario del libro.
4- Hacer que las imágenes sean ajustables por gusto propio, de tal manera que el escritor luzca su imagen de perfil, portada de libro o imágenes de los personajes como le plazca.
5- Añadir una gran biblioteca de fuentes para incluir en el editor de texto.
6- Posibilidad de añadir imágenes al editor de texto.
7- Añadir una papelera de reciclaje por si se desea rescatar libros que se han borrado.
8- Ampliar y ahondar en el apartado de análisis:
    a. Historial de errores con palabras más veces erradas, que también sea capaz de indicar si hay un exceso de muletillas.
    b. Palabras favoritas del escritor basándose en el uso que les da.
    c. Visualización de palabras clave.
    d. Lectura de diálogos e intervenciones de los personajes.
9- Realizar una exportación que incluya detalles de portada.
10- Realizar una importación desde pdf o Word que traduzca el libro y los separe en capítulos.
11- Proporcionar consejos de escritura más avanzados.
12- Hacer la aplicación adaptable a todos los dispositivos y pantallas.
13- Sincronizar los libros con la nube pudiendo hacer posible su guardado.
14- Realizar inicio de sesión con Google o Facebook.
15- Proporcionar la elección de la paleta de colores más oscura o clara en la pantalla de escritura, para facilitar la lectura y escritura del escritor.

9

Conclusión

Este proyecto ha supuesto una gran apertura al desarrollo como ingeniero. De principio a fin se ha estado debatiendo y compartiendo ideas, sacándole el máximo partido al enfoque grupal. ¿Por qué elegir tal tecnología? ¿Qué aporta frente a las demás? ¿Qué funcionalidades podrían añadirse al proyecto? ¿Qué partido le sacarían los usuarios finales, escritores? ¿Qué impacto tienen cada una de ellas? ¿Qué tipo base de datos se adapta mejor?

La búsqueda de la documentación, la resolución de conflictos de integración, la reestructuración y mejora del código… todos estos aspectos e inquietudes se han ido retroalimentando entre ambos ingenieros, cobrando gran motivación y altas perspectivas, generando creatividad y ganas de aprender y aplicar nuevos conceptos.

El conocimiento del lado cliente era prácticamente nulo antes de empezar, cada una de las tecnologías ha sido nueva, y, sin embargo, esto no ha supuesto impedimento. Se reafirma que la base adquirida durante el grado de ingeniería del software es suficiente, haciendo de los ingenieros una masa moldeable en cuanto aprendizaje, capaces de afrontar nuevos retos con gran flexibilidad y adaptabilidad.

Destacar uno de los puntos fuertes, uno de los grandes motivos de la ingeniería del software, como es el apoyo a otras doctrinas, el impulso de las ideas, la unión. Este aspecto es tan fundamental, que se decide crear una aplicación en base a ello. Con la libertad de elegir una idea que motive, y darle el impulso que se necesita para hacerla realidad con la ingeniería y el desarrollo software, se consigue ayudar al terreno de la escritura.

El desarrollo de esta aplicación web, supone pues un avance para ambos desarrolladores a nivel de conocimiento y visión de este ámbito, tal y como se esperaba. Se adquiere una toma de contacto con el desarrollo de proyectos a partir de una idea, favoreciendo el emprendimiento en un futuro y la aplicación de los distintos aspectos y métodos ingenieriles, teniendo en cuenta cada una de sus fases.

Gracias a este aspecto, se dota de un análisis de mercado, de una búsqueda de una necesidad al proyecto, lo cual supone un ingrediente más al aprendizaje adquirido.

La estimación en grupo, las veces que se tuvieron que adaptar las tareas por no poder cumplirlas a tiempo, es sino otro eslabón que hay que dominar y que se debía experimentar.

Todo esto, en resumidas cuentas, ha generado confianza; y contar con un entorno de trabajo totalmente interactivo, ha proporcionado comodidad y proactividad. El resultado puede verse en la aplicación, que responde a las necesidades del escritor en varios aspectos, con una ágil respuesta a nivel de interfaz debido a la integración de últimas tecnologías, y una buena consistencia a nivel de servicio gracias a una Api bien consolidada, intuitiva y fácil de mantener.

Con esto queda claro que el trabajo en equipo supone un motor hacia el éxito.

Referencias

¿Qué es NuGet y qué hace? (08 de enero de 2019). Obtenido de Microsoft Docs: https://docs.microsoft.com/es-es/nuget/what-is-nuget

Angular - Introduction to the Angular Docs. (10 de enero de 2019). Obtenido de Angular Docs: https://angular.io/docs

Angular | Font Awesome. (16 de enero de 2019). Obtenido de Font Awesome: https://fontawesome.com/how-to-use/on-the-web/using-with/angular

Angular powered Bootstrap. (04 de marzo de 2019). Obtenido de ng-bootstrap: https://ng-bootstrap.github.io/#/home

API REST: qué es y cuáles son sus ventajas en el desarrollo de proyectos. (18 de diciembre de 2018). Obtenido de BBVA API Market: https://bbvaopen4u.com/es/actualidad/api-rest-que-es-y-cuales-son-sus-ventajas-en-el-desarrollo-de-proyectos

Azure Cosmos DB: servicio de base de datos multimodelo. (03 de diciembre de 2018). Obtenido de Microsoft Azure: https://azure.microsoft.com/es-es/services/cosmos-db/

Blanco, N. (21 de marzo de 2019). *¿Qué patron usa Angular? MVC o MVVM*. Obtenido de OpenWebinars: https://openwebinars.net/blog/que-patron-usa-angular-mvc-o-mvvm/

Chacon, S. (12 de enero de 2019). *Git*. Obtenido de Git: https://git-scm.com/

Chart js | Open source HTML5 for your website . (01 de abril de 2019). Obtenido de Chart.js: https://www.chartjs.org/

Documentación de ASP.NET. (15 de noviembre de 2018). Obtenido de Microsoft: https://docs.microsoft.com/es-es/aspnet/?view=aspnetcore-2.2#pivot=core

Explicación de DevOps: Información general sobre las tecnologías de DevOps. (s.f.). Obtenido de Microsoft Azure: https://azure.microsoft.com/es-es/overview/devops/

Guía de C#. (02 de octubre de 2018). Obtenido de Microsoft Docs: https://docs.microsoft.com/es-es/dotnet/csharp/

Introducción a las consultas LINQ (C#). (18 de diciembre de 2018). Obtenido de Microsoft Docs: https://docs.microsoft.com/es-es/dotnet/csharp/programming-guide/concepts/linq/introduction-to-linq-queries

Introduction Bootstrap. (11 de enero de 2019). Obtenido de Bootstrap: https://getbootstrap.com/docs/4.3/getting-started/introduction/

JSON. (05 de enero de 2019). Obtenido de Wikipedia: https://es.wikipedia.org/wiki/JSON

JSON Web Tokens. (05 de enero de 2019). Obtenido de JWT: https://jwt.io/

Json.NET. (19 de enero de 2019). Obtenido de Newtonsoft: https://www.newtonsoft.com/json

MongoDB: The most popular database for modern apps. (04 de febrero de 2019). Obtenido de MongoDB: https://www.mongodb.com/

npm Documentation. (25 de enero de 2019). Obtenido de npm Documentation: https://docs.npmjs.com/

Quickstart - Quill Rich Text Editor. (02 de febrero de 2019). Obtenido de Quilljs: https://quilljs.com/docs/quickstart/

Scrum (desarrollo de software). (10 de marzo de 2019). Obtenido de Wikipedia: https://es.wikipedia.org/wiki/Scrum_(desarrollo_de_software)

Spec India. (14 de enero de 2019). *React vs Angular vs Vue.js: A Complete Comparison Guide*. Obtenido de Medium: https://medium.com/front-end-weekly/react-vs-angular-vs-vue-js-a-complete-comparison-guide-d16faa185d61

SWAGGER Y SWAGGER UI: ¿QUÉ ES Y POR QUÉ ES IMPRESCINDIBLE PARA TUS APIS? (s.f.). Obtenido de Chakray: https://www.chakray.com/es/swagger-y-swagger-ui-por-que-es-imprescindible-para-tus-apis/

Transferencia de Estado Representacional. (18 de noviembre de 2018). Obtenido de Wikipedia: https://es.wikipedia.org/wiki/Transferencia_de_Estado_Representacional

TypeScript. (10 de enero de 2019). Obtenido de Wikipedia: https://es.wikipedia.org/wiki/TypeScript

Valor Software. (05 de marzo de 2019). *NGX Bootstrap*. Obtenido de NGX Bootstrap: https://valor-software.com/ngx-bootstrap/#/documentation

Visual Studio Documentación. (01 de octubre de 2018). Obtenido de Microsoft Docs: https://docs.microsoft.com/es-es/visualstudio/?view=vs-2017

Anexo: Guía de uso para usuario

En este apartado se explicarán los pasos que los usuarios han de seguir para usar la aplicación web, adjuntando para ello imágenes que ilustren y clarifiquen el recorrido por la misma.

Acceso a la aplicación

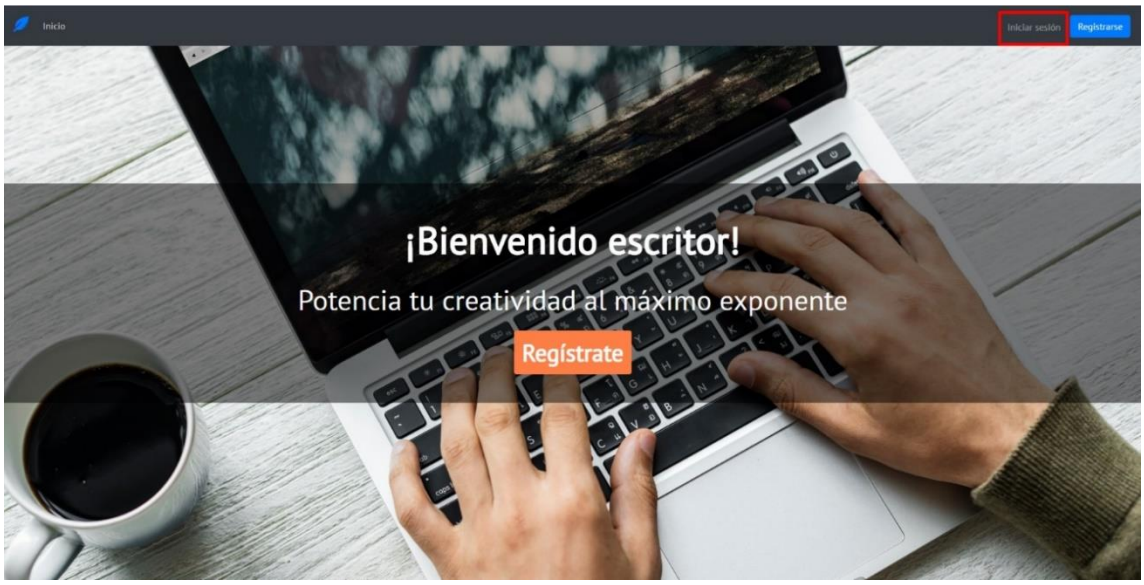


Figura 41 – Página de bienvenida a la aplicación

Para usuarios registrados: en la pestaña de inicio, pulsar el botón “Iniciar sesión”.

Entonces se abre una ventana donde introducir los datos de acceso: correo electrónico y contraseña, ambos obligatorios. Tras introducirlos, pulsar el botón “Iniciar sesión”.

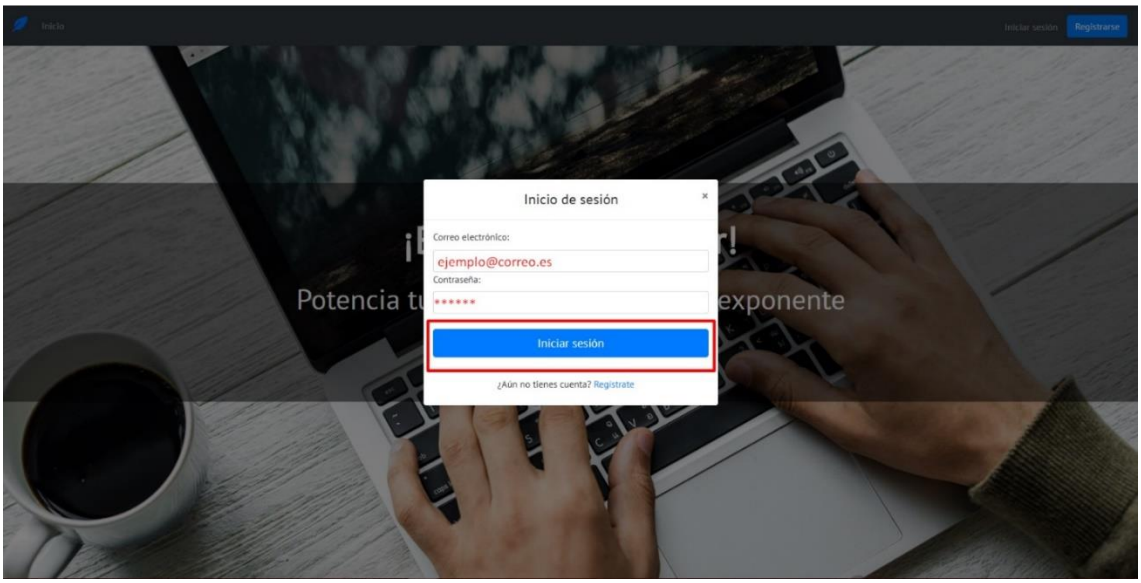


Figura 42 – Página de inicio de sesión

Para usuarios no registrados: en la pestaña de inicio, pulsar alguno de los dos botones de registro.

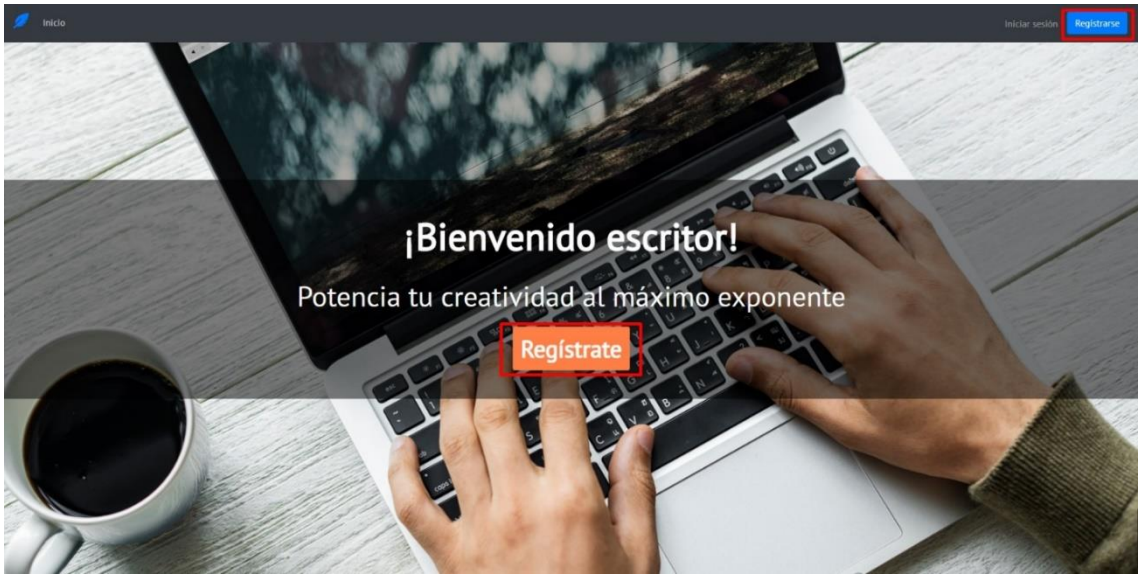


Figura 43 – Página de bienvenida, botones para registrarse

Se abre ventana de registro, todos los campos son obligatorios, y el botón “registrarme” aparece desactivado hasta que se rellenen los campos.

Una vez completados, pulsar el botón. Tras ello, deberá iniciar sesión introduciendo los datos de accesos “correo electrónico” y “contraseña”.

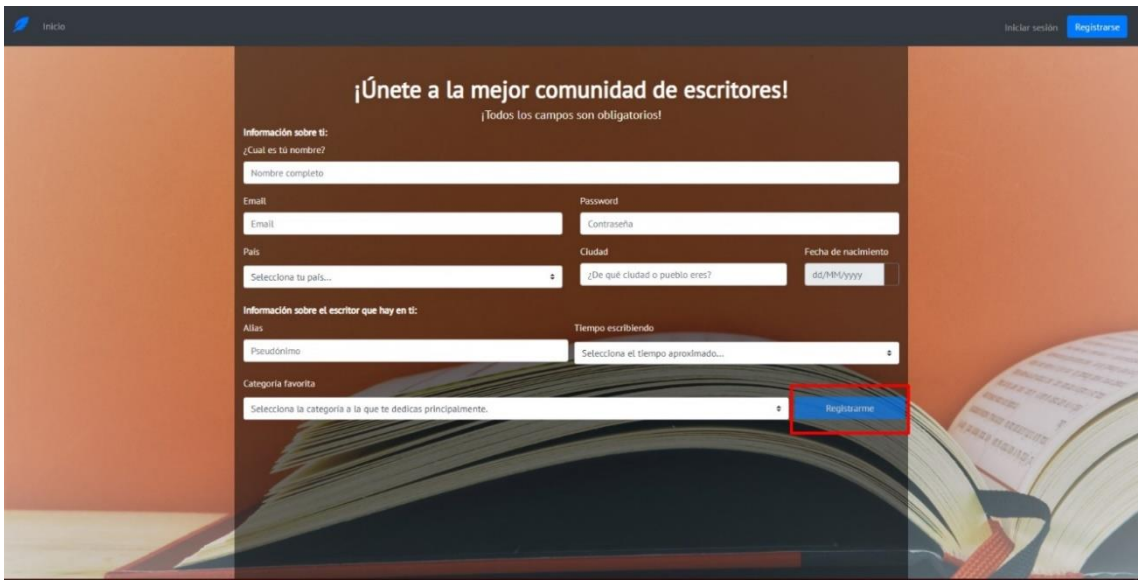


Figura 44 – Página de registro de usuarios

Cerrar sesión

Una vez se accede a la aplicación, arriba a la derecha en la barra de navegación, aparece un desplegable, al pulsar, se visualiza la opción de “cerrar sesión”. Pulsar sobre ella.

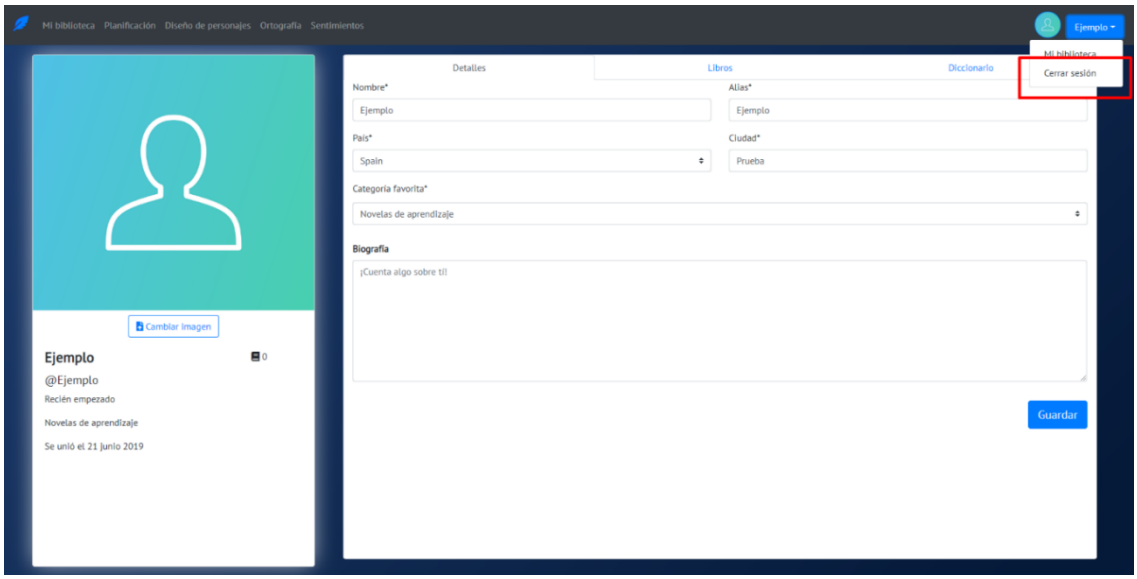
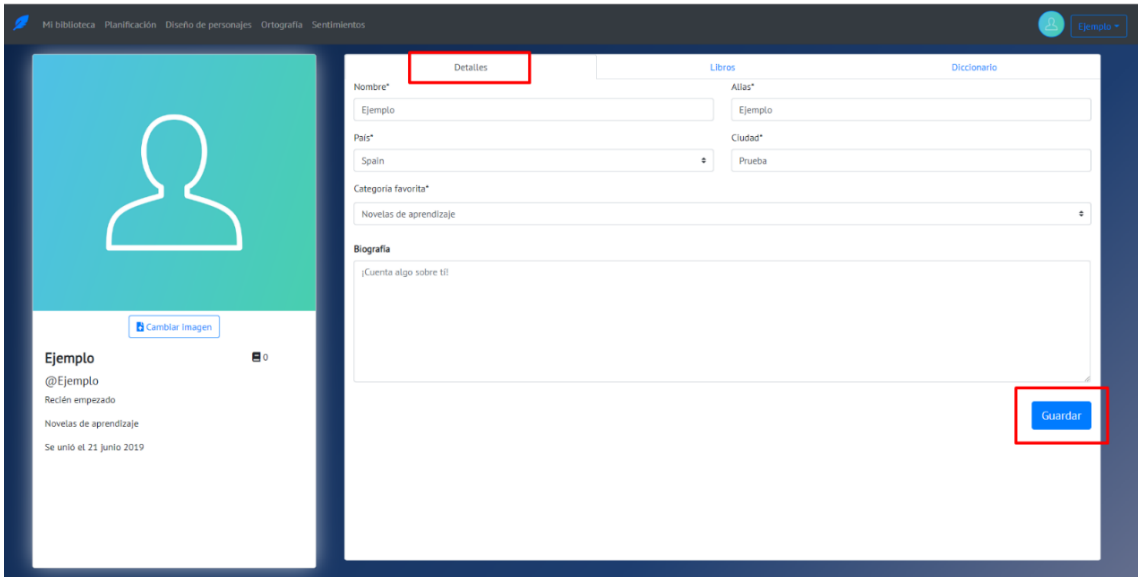


Figura 45 – Acción de cerrar sesión de la aplicación

Editar datos del perfil

En la pestaña de mi biblioteca, se visualiza los datos del perfil del escritor junto a sus libros. Pulsar sobre detalles y rellenar los datos que se desean modificar. No se puede dejar en blanco ninguno que esté marcado con asterisco.

Figura 46 – Perfil del escritor, detalles del escritor



Si se desea cambiar la imagen del perfil, pulsar sobre “Cambiar imagen”.

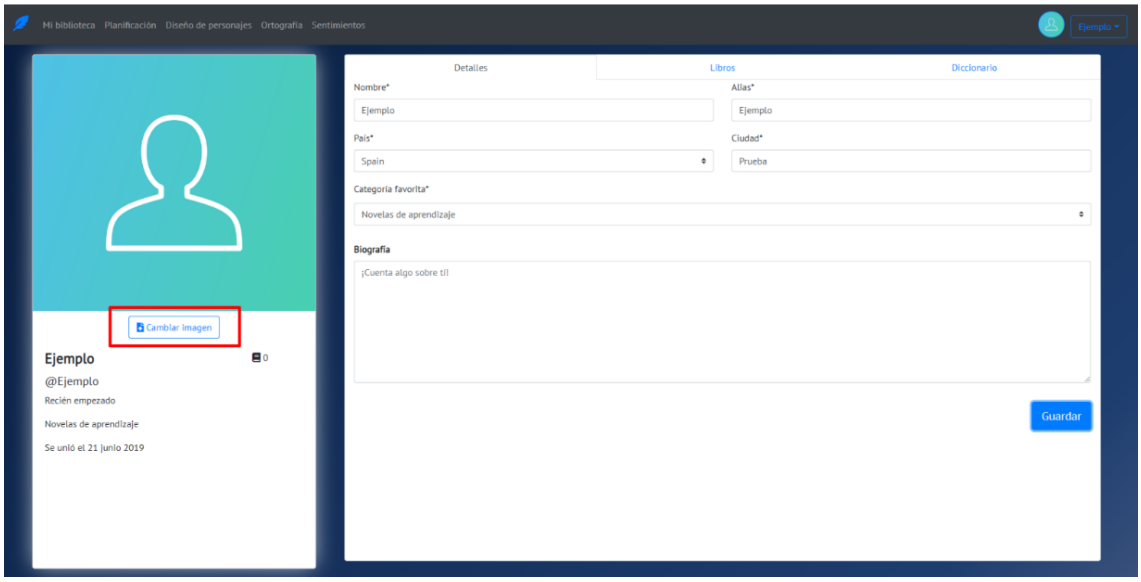


Figura 47 – Subida de imagen en el perfil del escritor

Al hacerlo se abrirá el explorador de archivos del sistema operativo. Seleccionar una imagen.

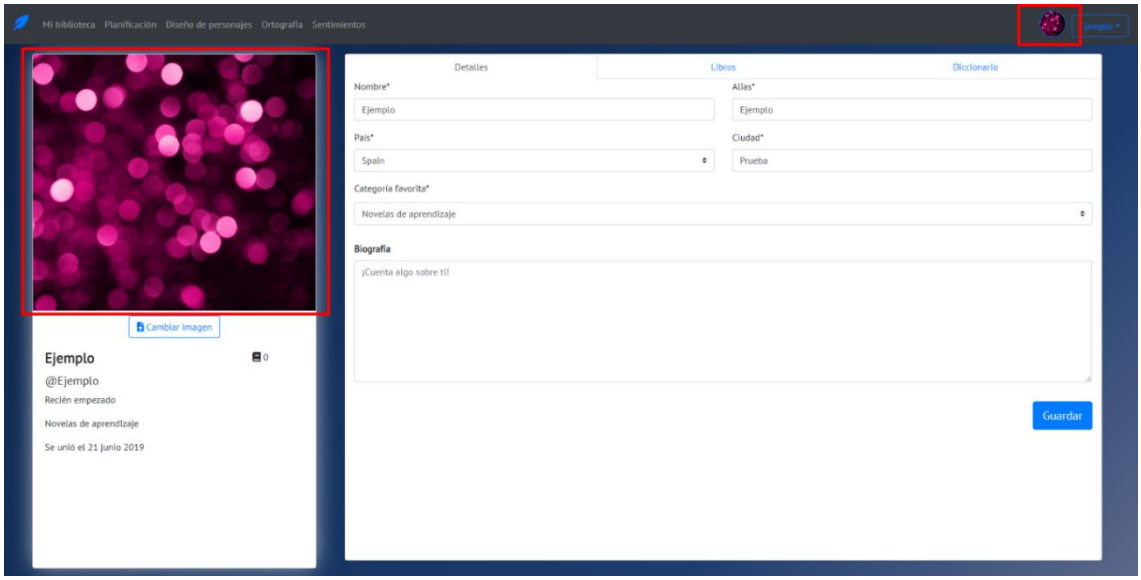


Figura 48 – Perfil del usuario con imagen cambiada

Planificación de tareas

Seleccionar “Planificación” en la barra de navegación.

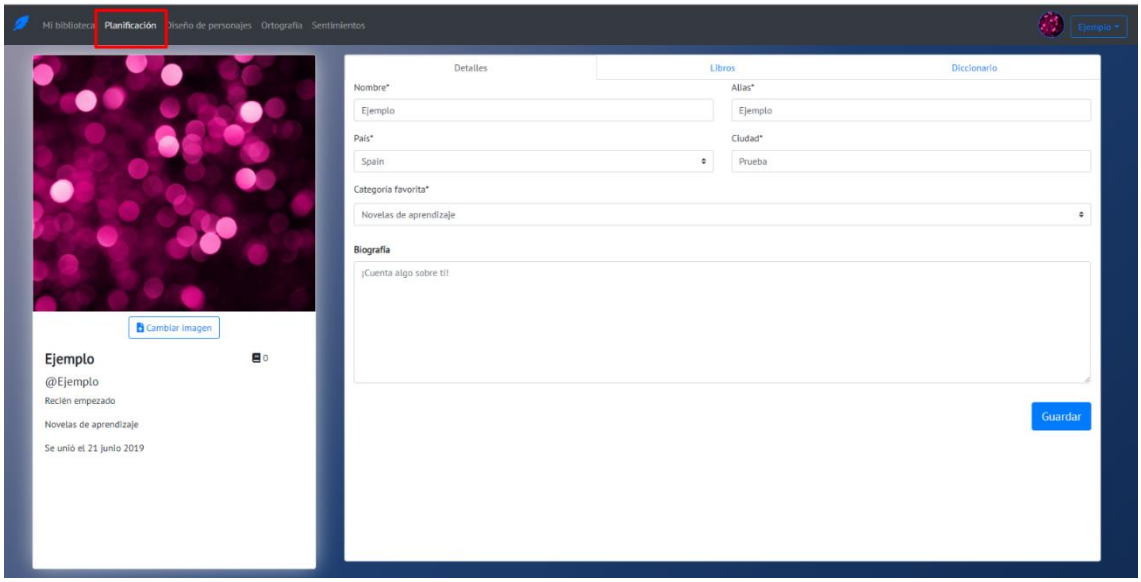


Figura 49 – Navegar desde perfil del escritor al planificador de código

Una vez dentro, crear una tarea pulsando el botón **+**

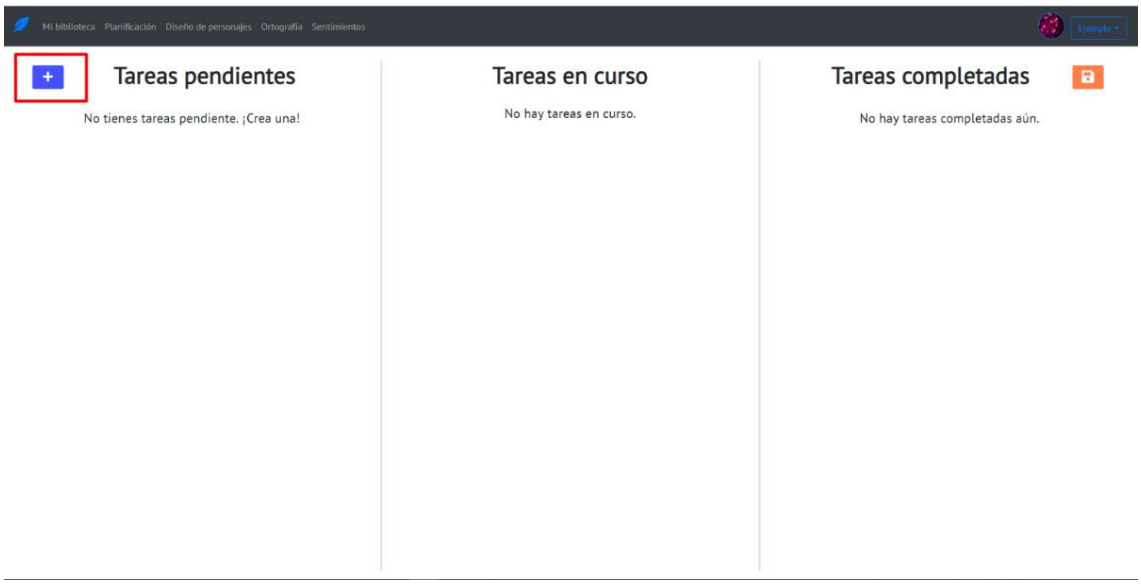


Figura 50 – Página de planificador de tareas

Se abrirá una ventana para rellenar el título de la tarea, la descripción y la fecha esperada de cumplimiento. El título y la fecha son obligatorios.

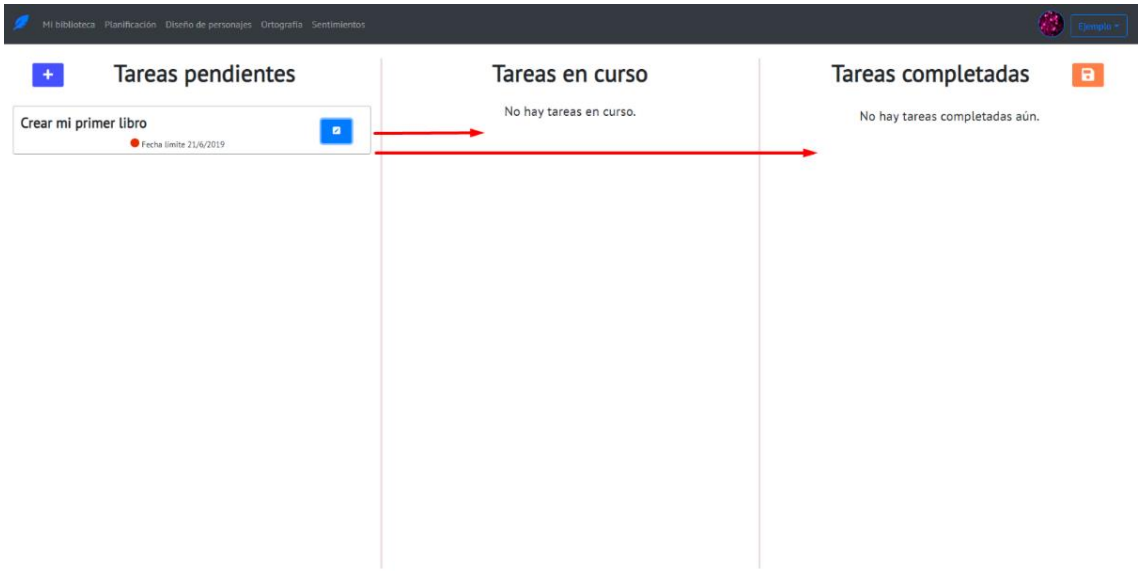


Figura 51 – Planificador de tareas, acción de mover tarea pendiente

Se creará en tareas pendientes, y podrá ser arrastrada a las demás columnas.

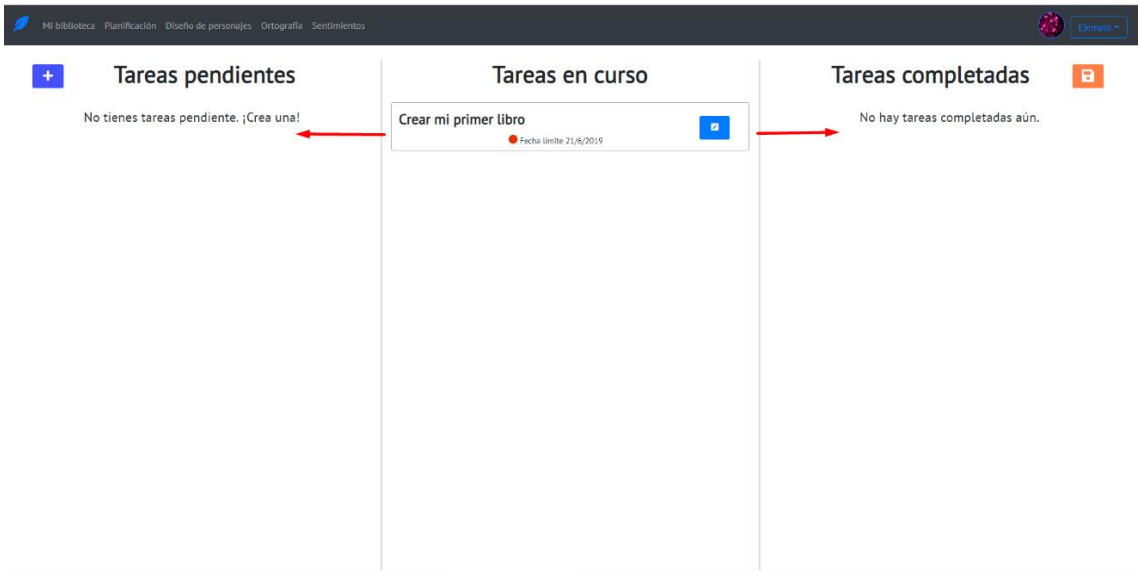


Figura 52 – Planificador de tareas, acción de mover tarea en curso

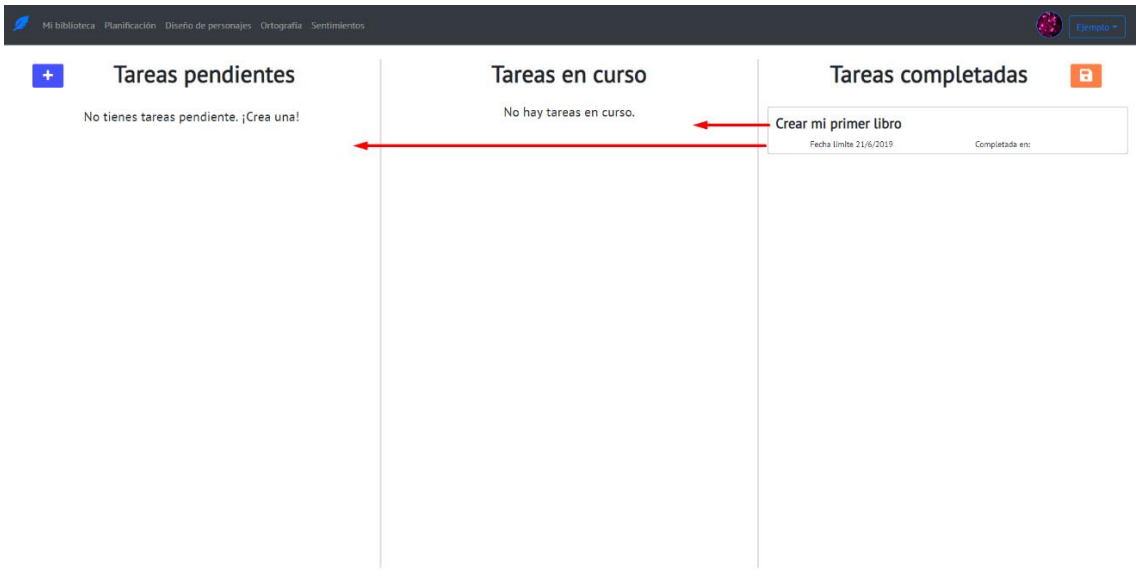


Figura 53 – Planificador de tareas, acción de mover tarea completada

Para editar el estado de la tarea pulsar el botón de edición.

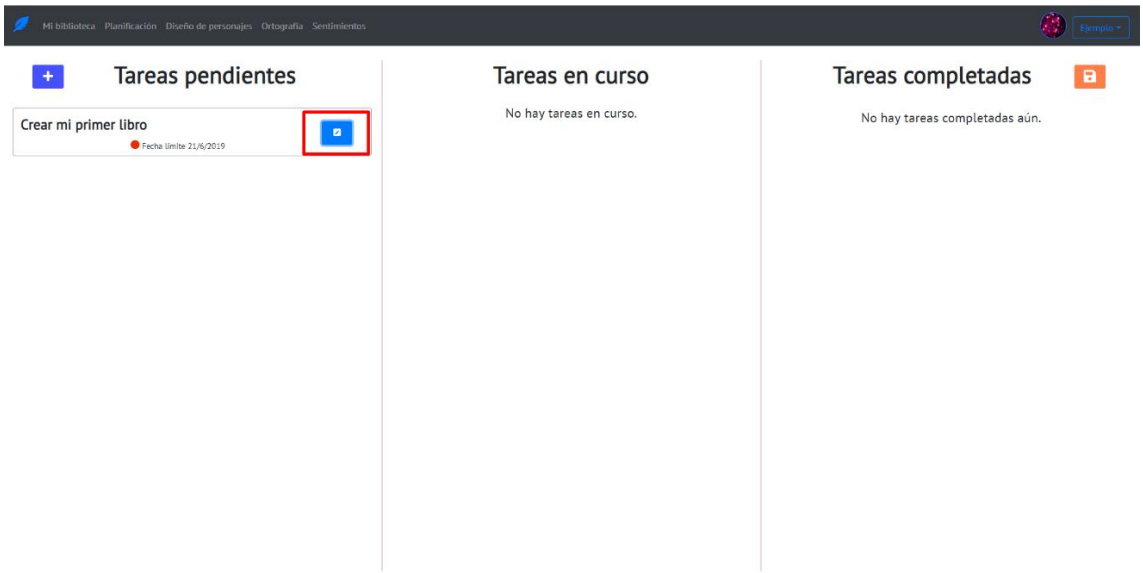


Figura 54 – Planificador de tareas, acción de mover tarea en curso

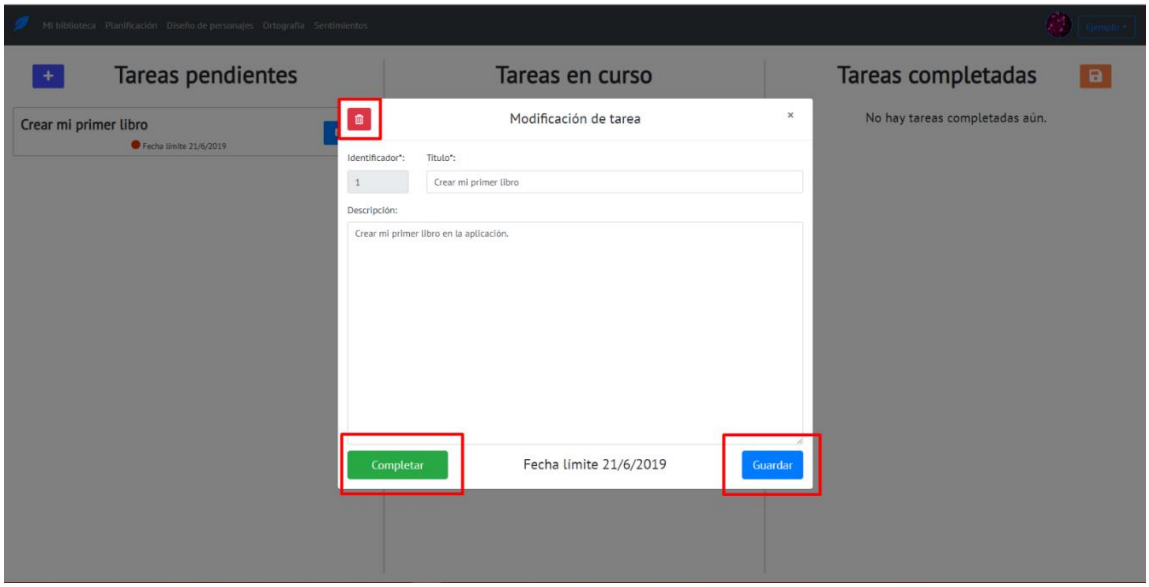


Figura 55 – Planificador de tareas, modificar tarea

Tras pulsar, se visualiza una ventana de edición. Se podrá borrar, editar, o completar la tarea.

Al pulsar en el botón de la papelera se abre una ventana de confirmación para borrar, seleccionar “Eliminar” si realmente se desea borrar la tarea.

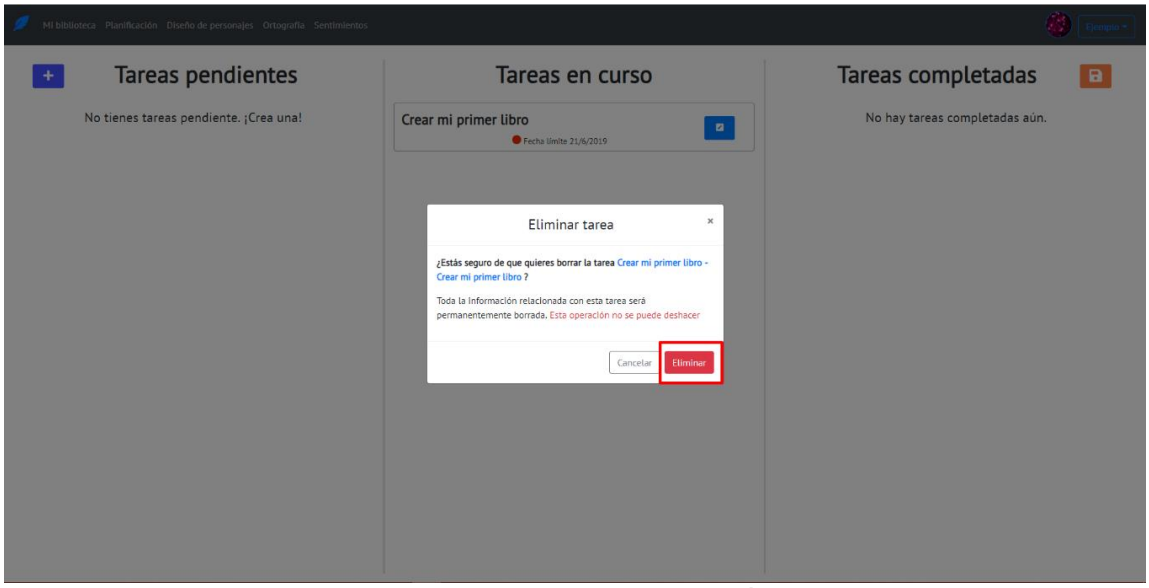


Figura 56 – Planificador de tareas, acción de eliminar tarea

Pulsar el botón “Guardar”, para confirmar los nuevos cambios sobre la tarea.

Pulsar el botón “Completar” si lo que desea es poner la tarea en la columna “Tareas completadas”.

Para que no se pierdan los cambios y el estado final del panel de tareas, pulsar sobre el botón de guardado de arriba a la derecha.

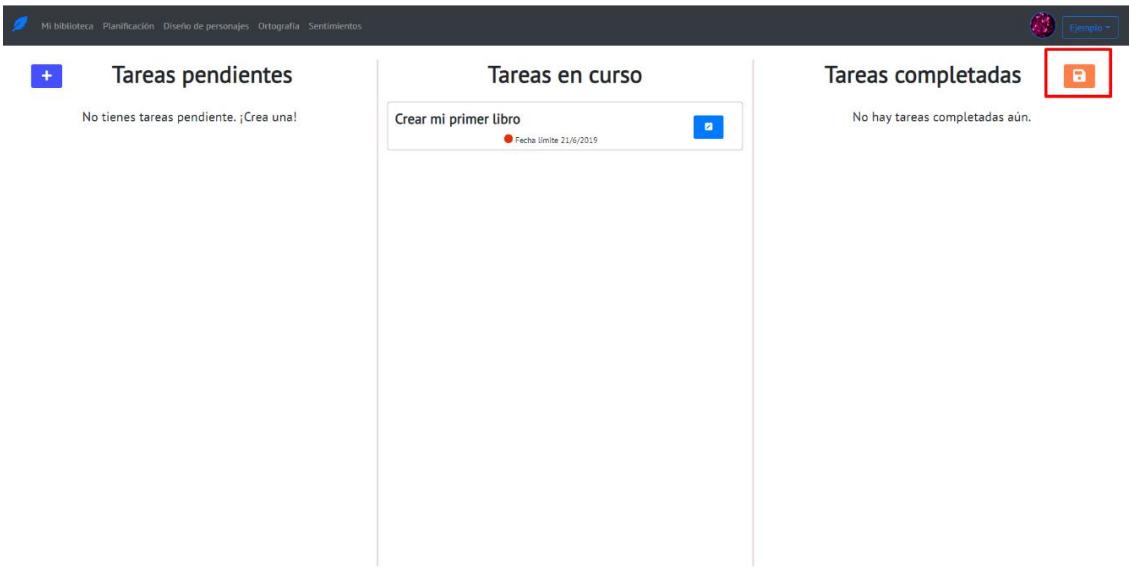


Figura 57 – Planificador de tareas, acción de guardar cambios

Crear, borrar editar palabra en el diccionario

Desde la pantalla Mi biblioteca, seleccionar la pestaña diccionario, desde ahí, pulsar “Nueva palabra”.

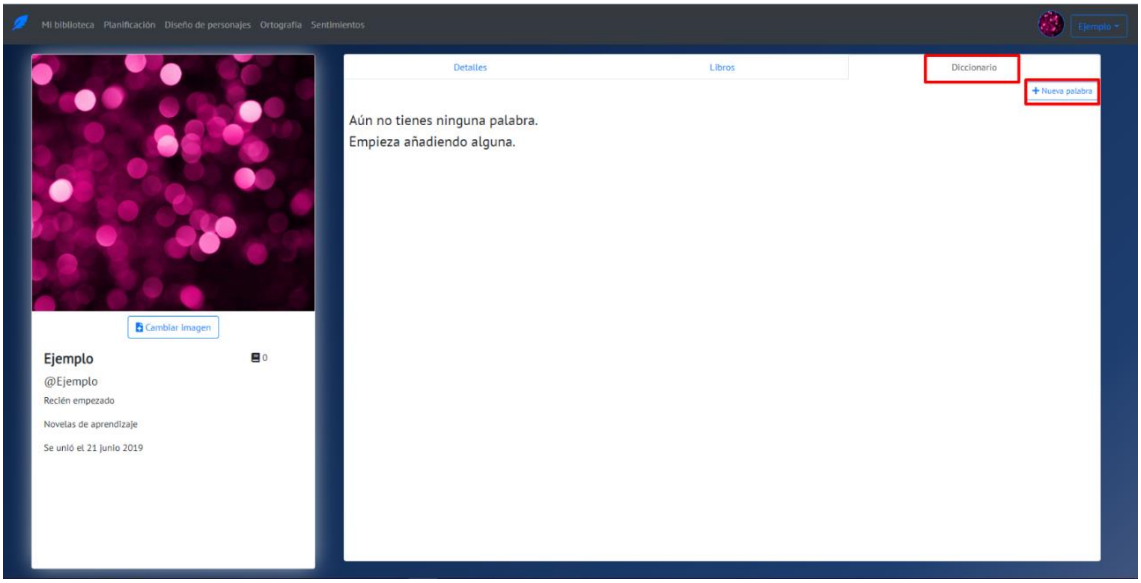


Figura 58 – Página del diccionario del escritor

Al hacerlo aparece una ventana de creación donde rellenar la palabra y su descripción. El campo “palabra” es obligatorio y hasta que no se rellene no se podrá crear la palabra.

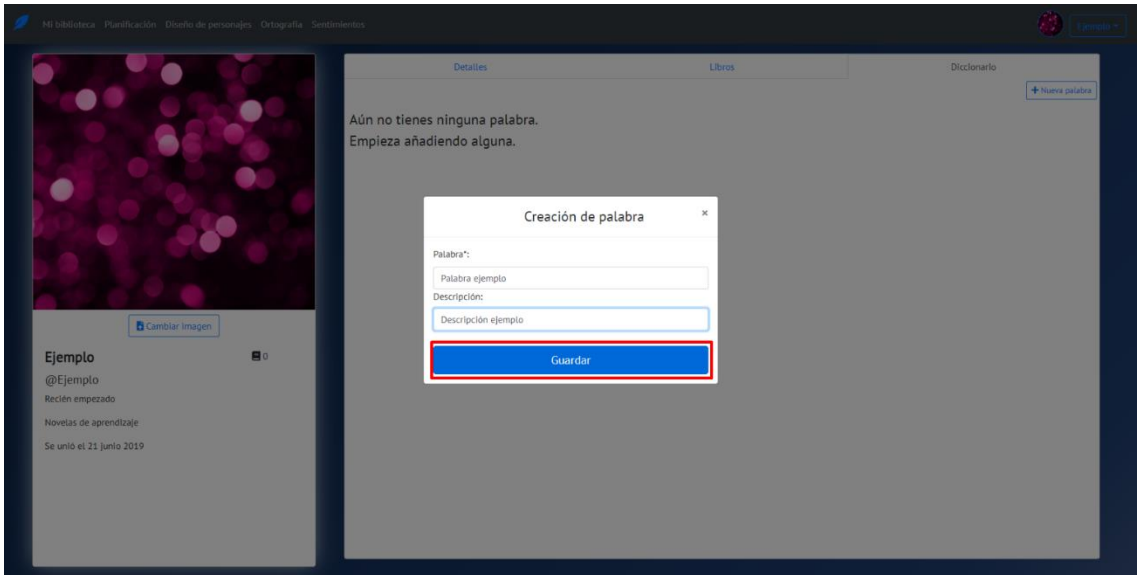


Figura 59 – Formulario de creación de palabra del diccionario

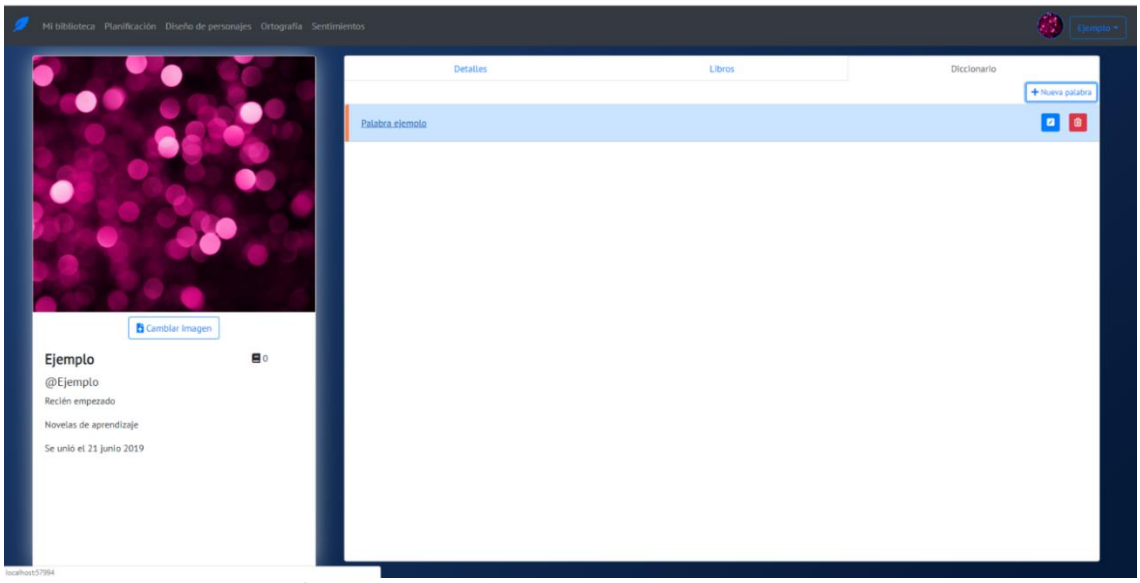


Figura 60 – Página del diccionario del escritor con palabra añadida

Para visualizar su descripción, pulsar sobre la palabra en la lista.

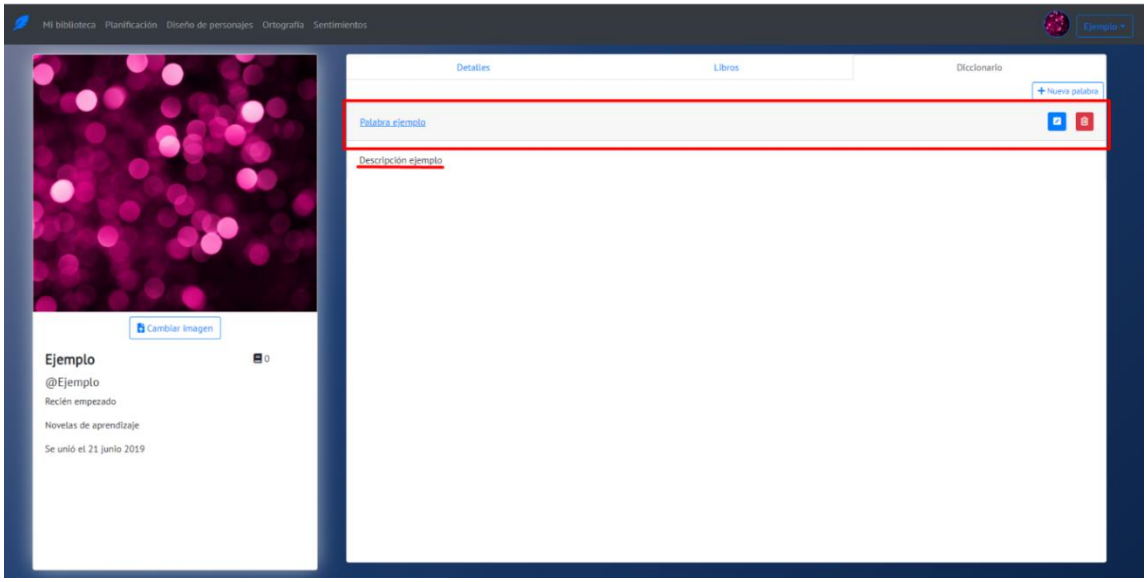


Figura 61 – Diccionario del escritor acción pinchar sobre palabra

Para editar, o borrar, pulsar los botones situados a la derecha respectivamente.

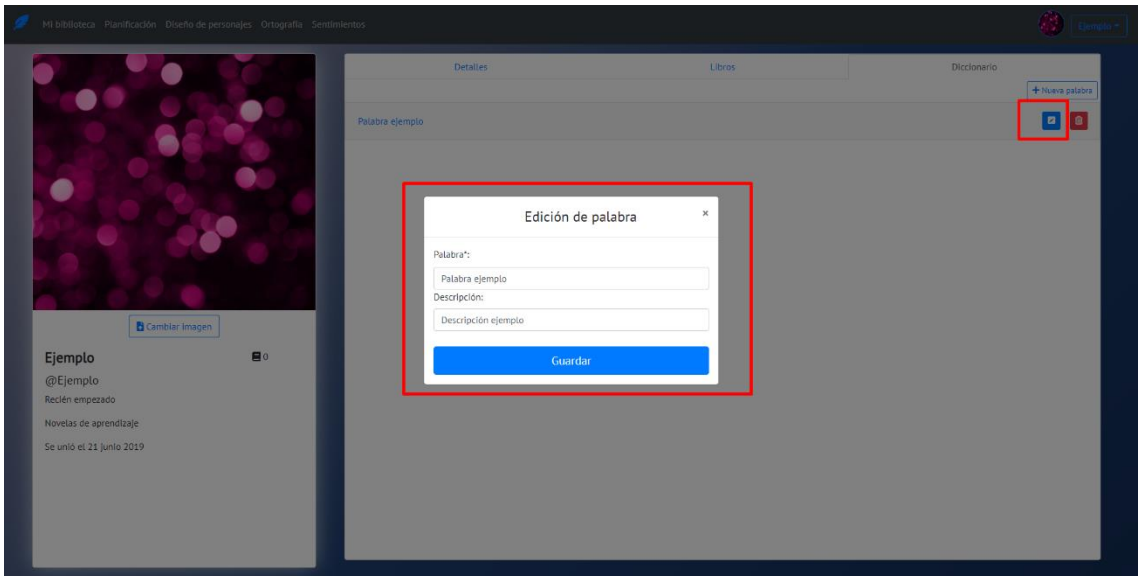


Figura 62 – Diccionario del escritor, formulario de modificación de palabra

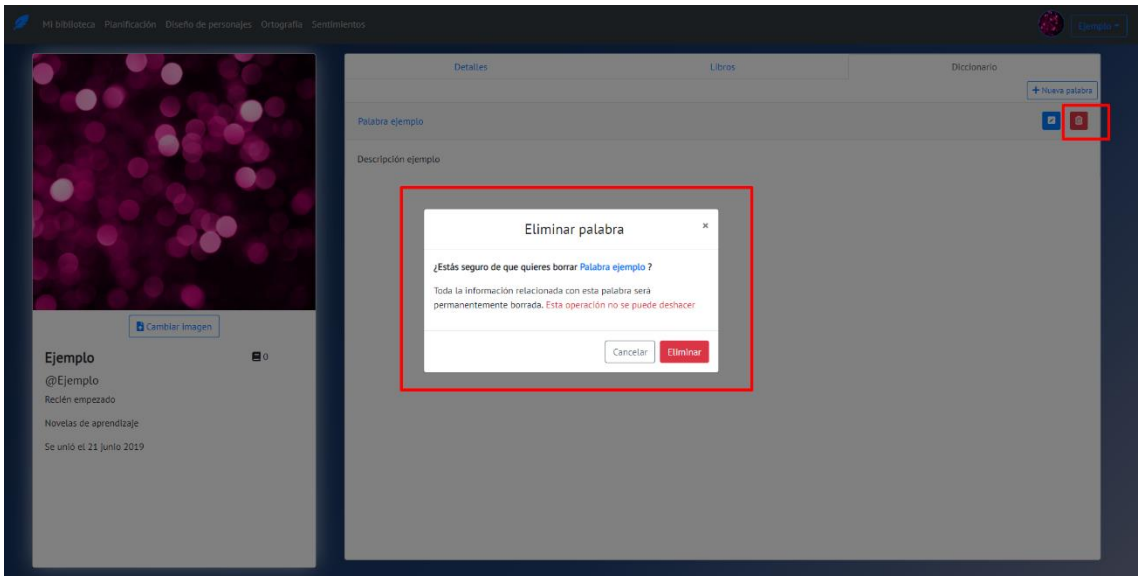


Figura 63 – Acción de eliminar una palabra del diccionario

Crear y borrar libro

Desde la pantalla de mi biblioteca, seleccionar la pestaña libros y pulsar “Nuevo libro”.

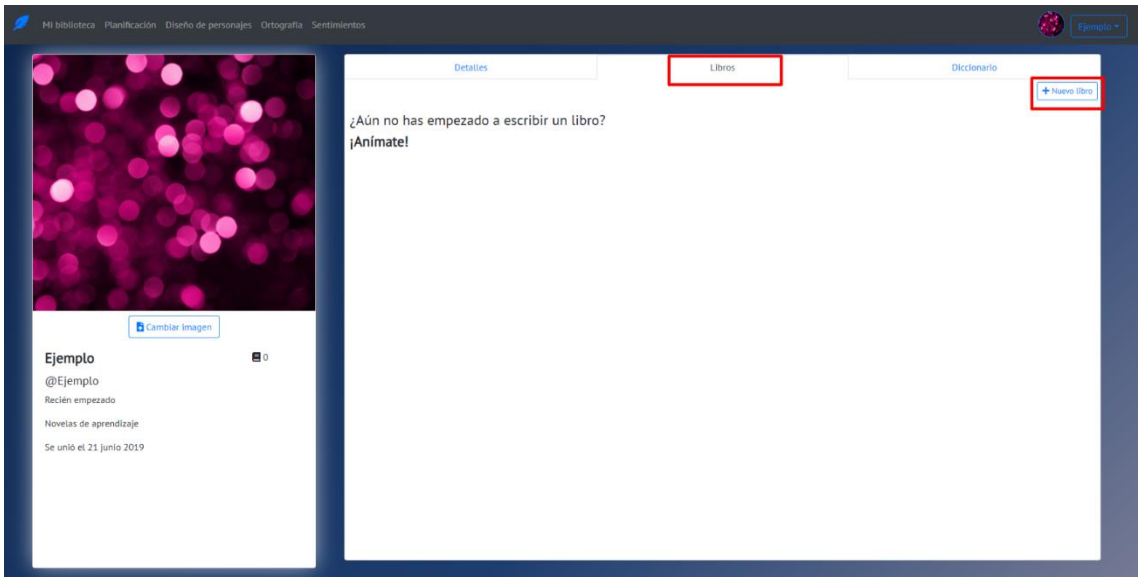


Figura 64 – Página del listado de libros del escritor

Al hacerlo aparece la ventana de creación. Ningún campo es obligatorio, se puede crear el libro al pulsar el botón crear libro.

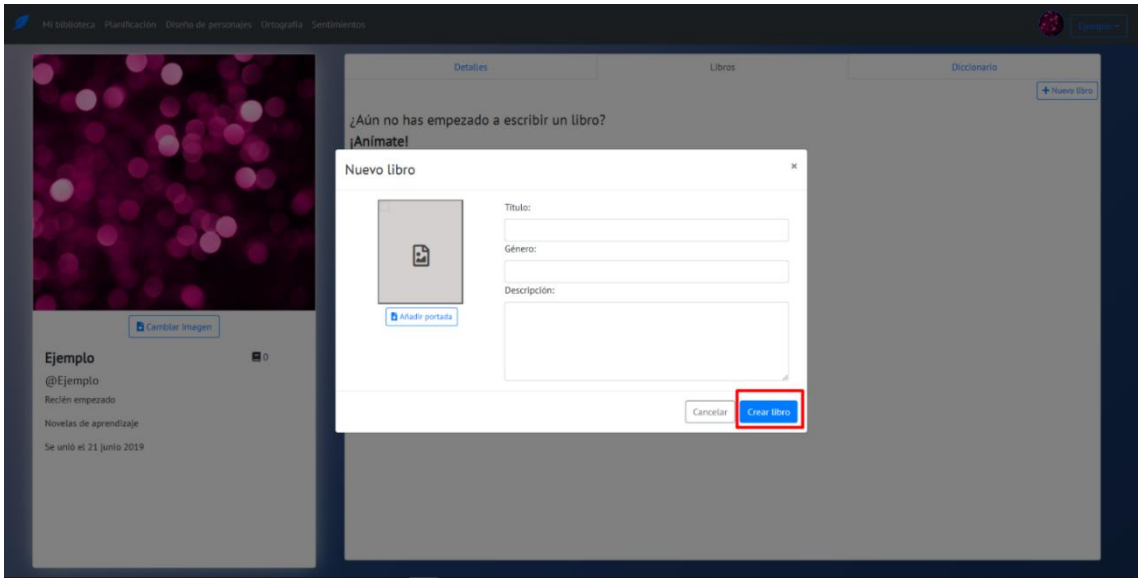


Figura 65 – Página creación de libro nuevo

Si se desea subir la portada al momento de creación, pulsar “Añadir portada”. Se abre explorador de archivos y se podrá elegir una imagen. Al seleccionarla se visualiza.

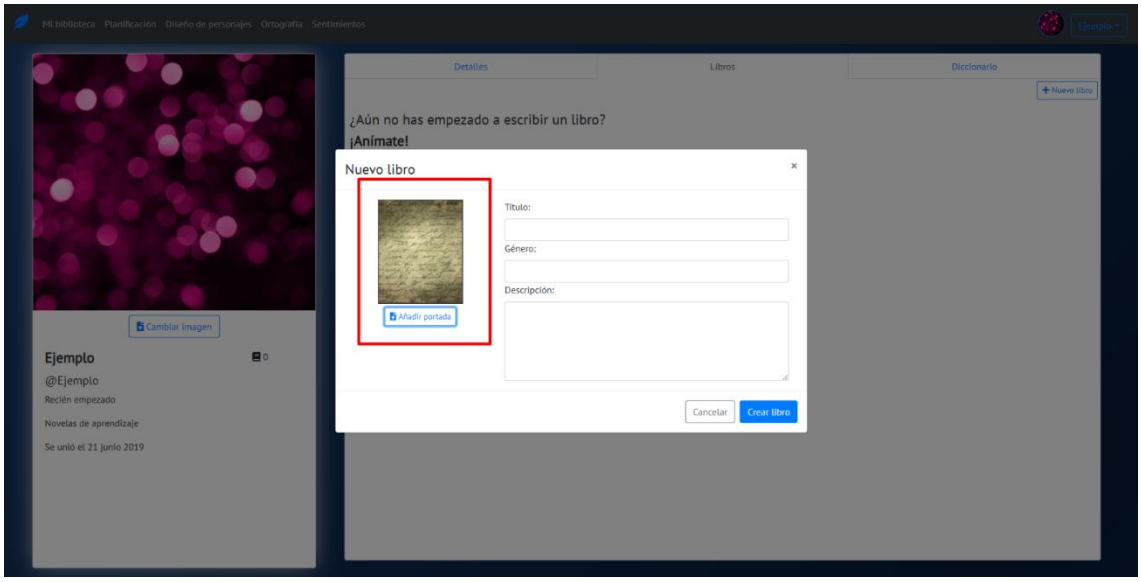


Figura 66 – Subida de imagen de la portada del libro

Al pulsar botón “Crear libro”, se accede directamente a su pantalla de detalles. Otra forma de acceder es pulsando sobre el libro creado en la lista.

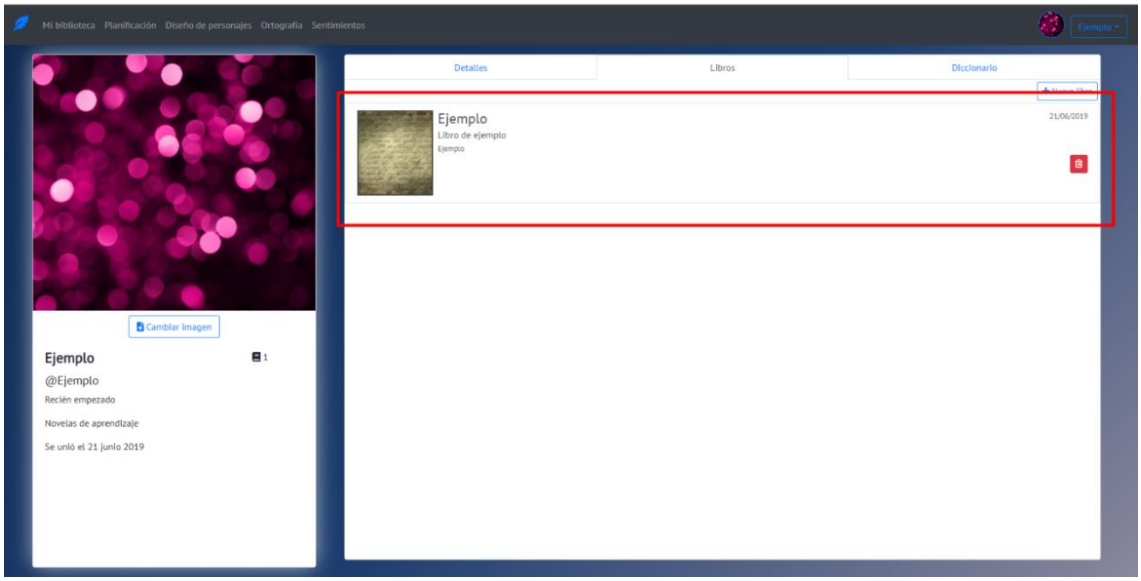


Figura 67 – Página del listado de libros

Si se desea borrar el libro, pulsar el botón rojo de la papelera y dar al botón de confirmar una vez aparezca la pestaña de confirmación.

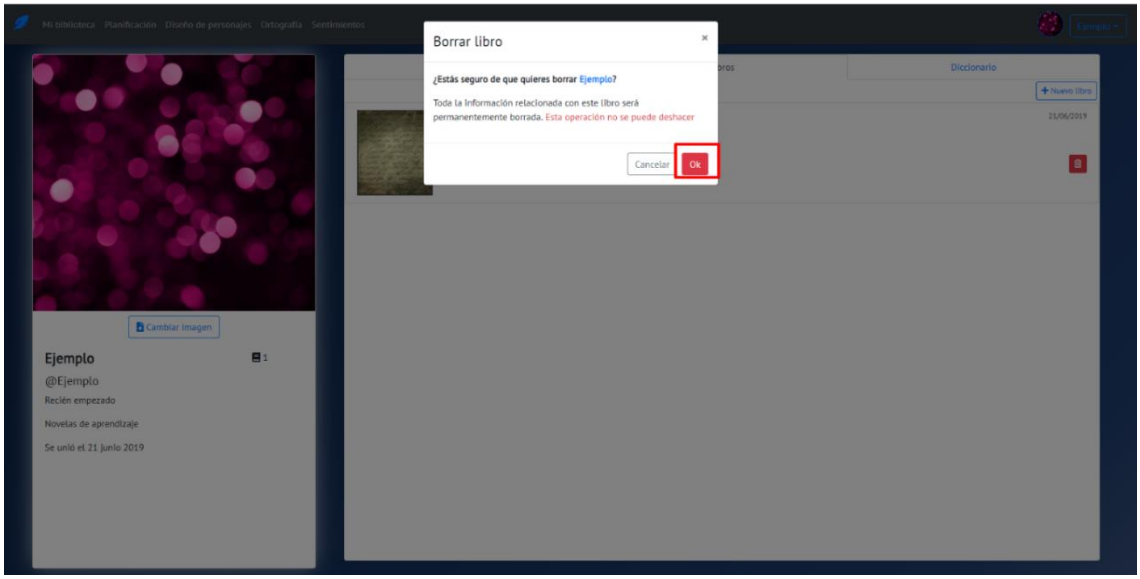


Figura 68 – Página de la eliminación de un libro

Editar detalles de libro

Desde la pantalla de detalles del libro, pulsar la pestaña “Detalles”. Ningún campo es obligatorio, rellenarlos y pulsar el botón “Guardar”. Si se desea cambiar la imagen de la portada, pulsar sobre el botón “Cambiar portada”, y seleccionar imagen.

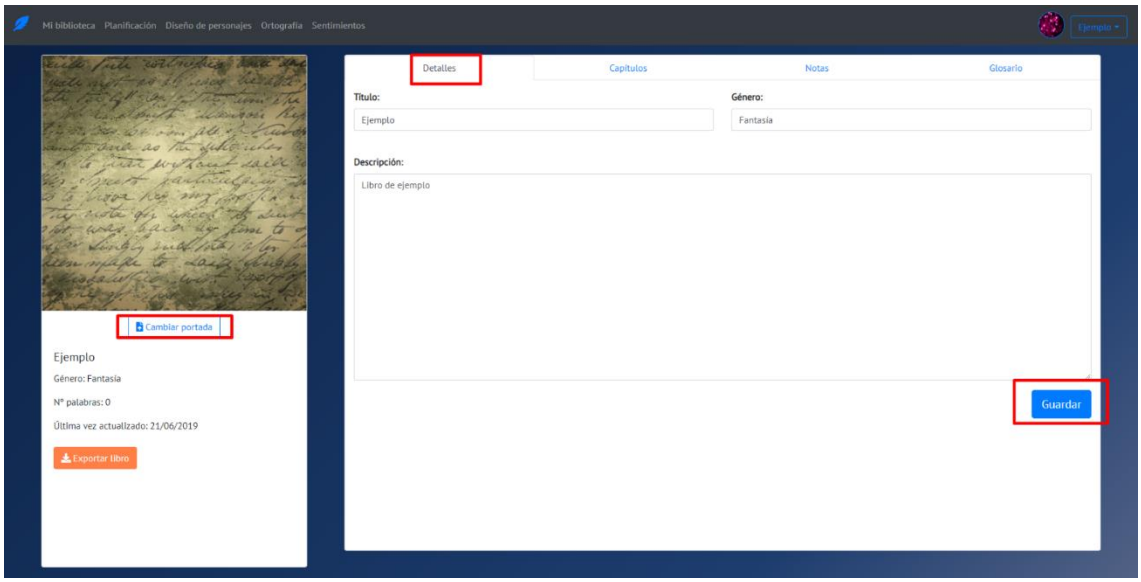


Figura 69 – Página de detalles del escritor

Crear, editar y borrar palabra glosario

Este apartado tiene el mismo funcionamiento que la creación de palabras del diccionario personal del escritor. Acceder a la pestaña correspondiente, “Glosario”, seleccionar “Nueva palabra”, pulsar sobre ella una vez creada para ver el significado, y pulsar sobre los botones de edición o borrado en caso de necesitarlo.

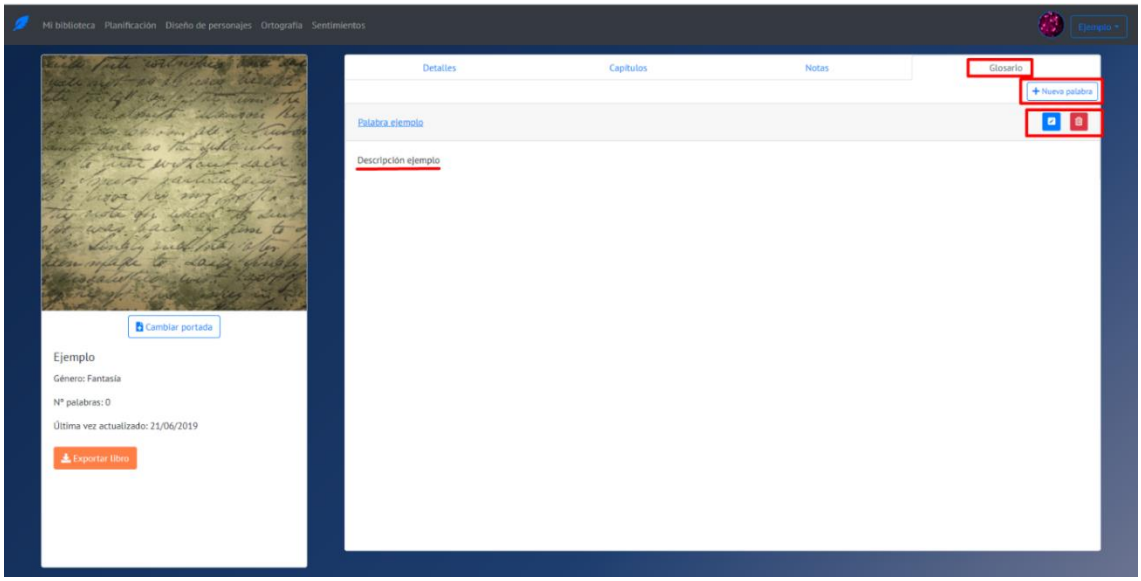


Figura 70 – Página del glosario de un libro

Crear editar y borrar nota

De nuevo, el conjunto de pasos es el mismo que los vistos en diccionario y glosario, la única diferencia es que no hay botón de editado, basta con pulsar sobre el elemento una vez creado, ya que la información se visualiza desde fuera.

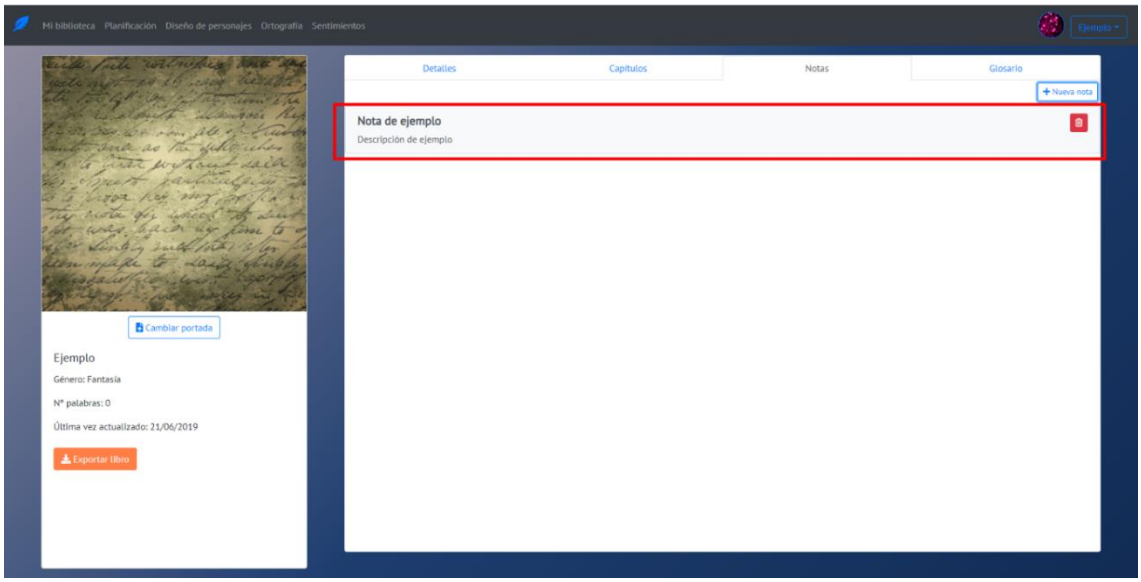


Figura 71 – Página del listado de notas de un libro

Si la descripción es demasiado larga, se puede ajustar el tamaño de entrada, arrastrando desde la esquina inferior derecha el recuadro de texto.

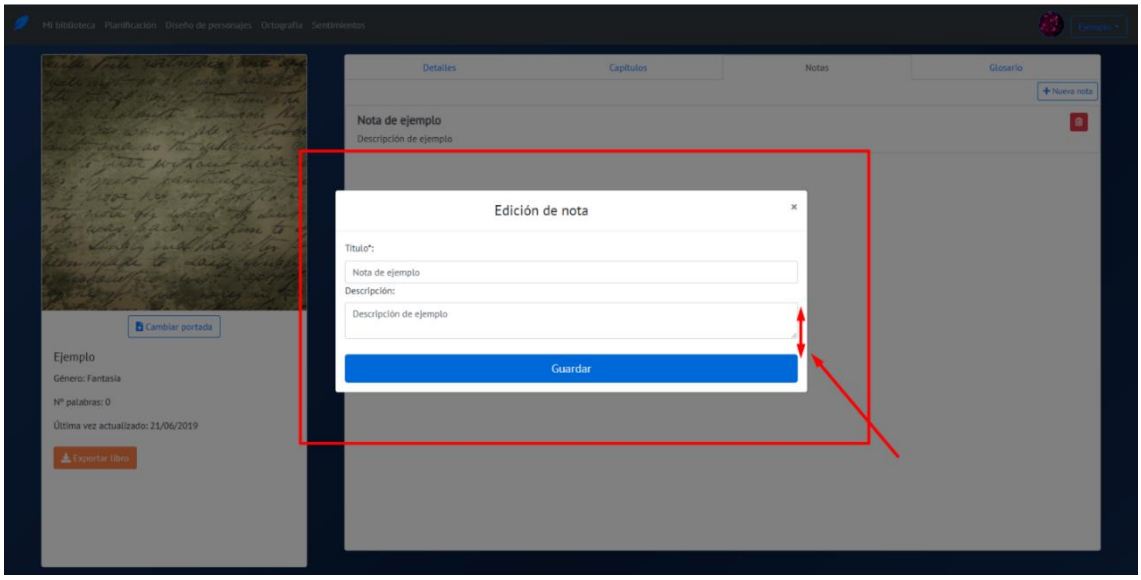


Figura 72 – Página de modificación de una nota

Crear y borrar capítulos

Desde la pantalla de detalles del libro, pulsar la pestaña “Capítulos”, y pulsar el botón “Nuevo Capítulo”.

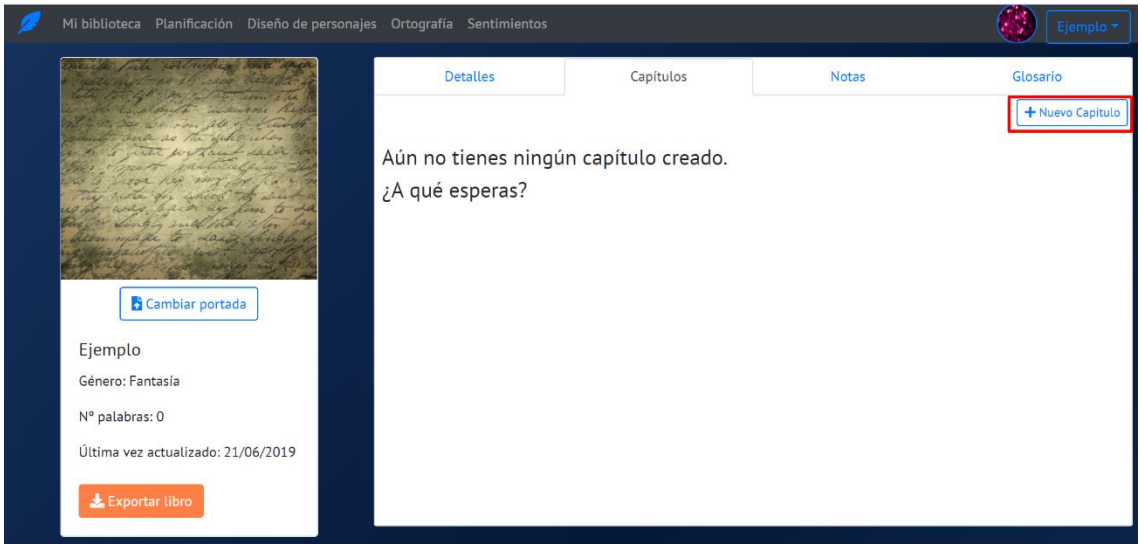


Figura 73 – Página de capítulos de un libro

Al hacerlo redireccionará a la pantalla de escritura en línea.

Otra forma de acceder a ella es pulsando sobre el creado al volver atrás.

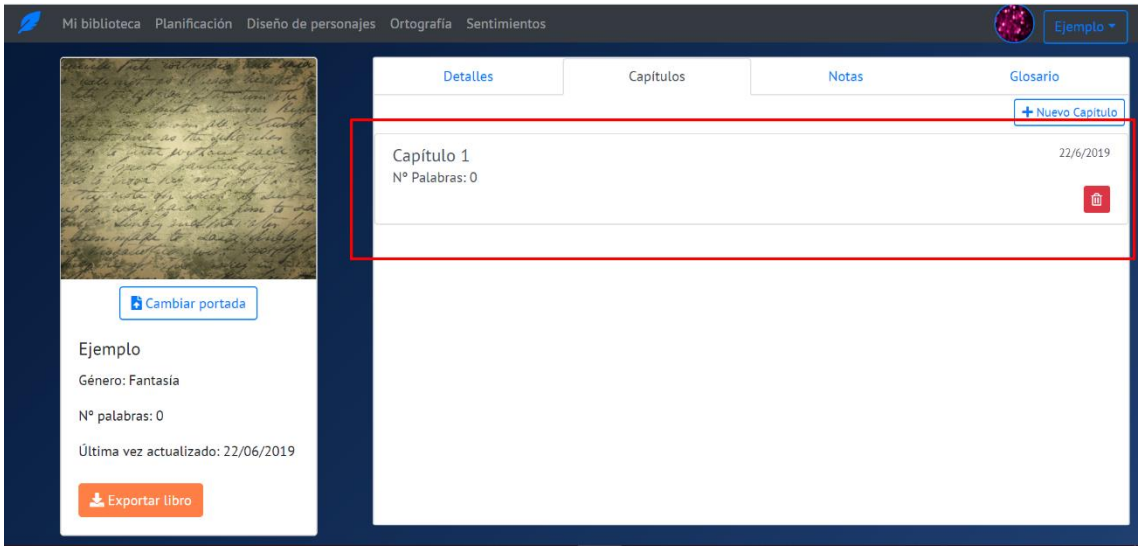


Figura 74 – Acciones sobre los capítulos de un libro

Para borrar el capítulo, pulsar sobre la papelera roja, y confirmar el borrado en la ventana de confirmación.

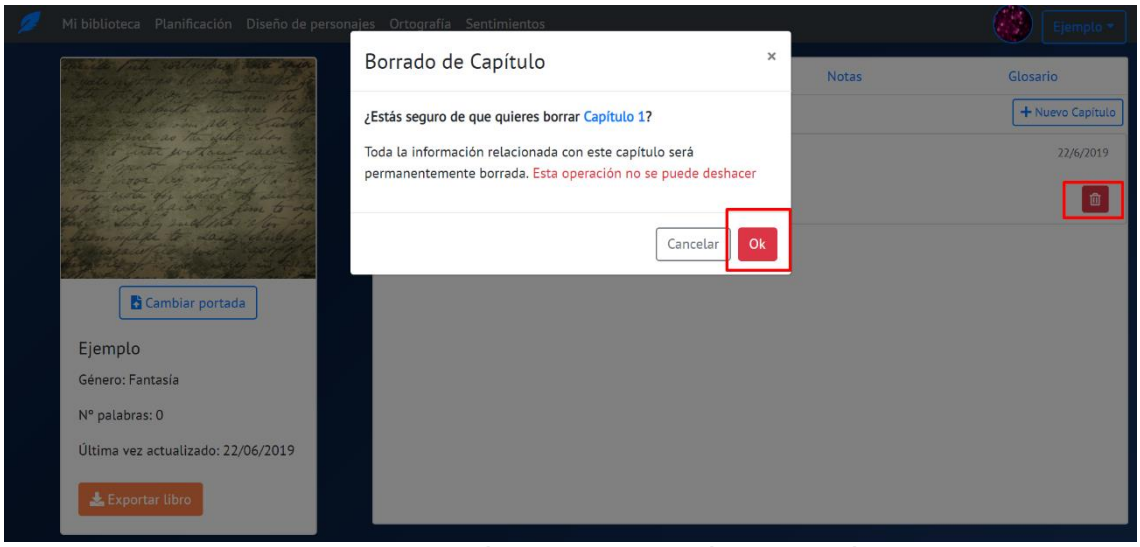


Figura 75 – Página eliminación de capítulo

Escribir Capítulo

Desde la pantalla de escritura en línea, se visualiza el editor de texto. Escribir sobre él el contenido del capítulo.

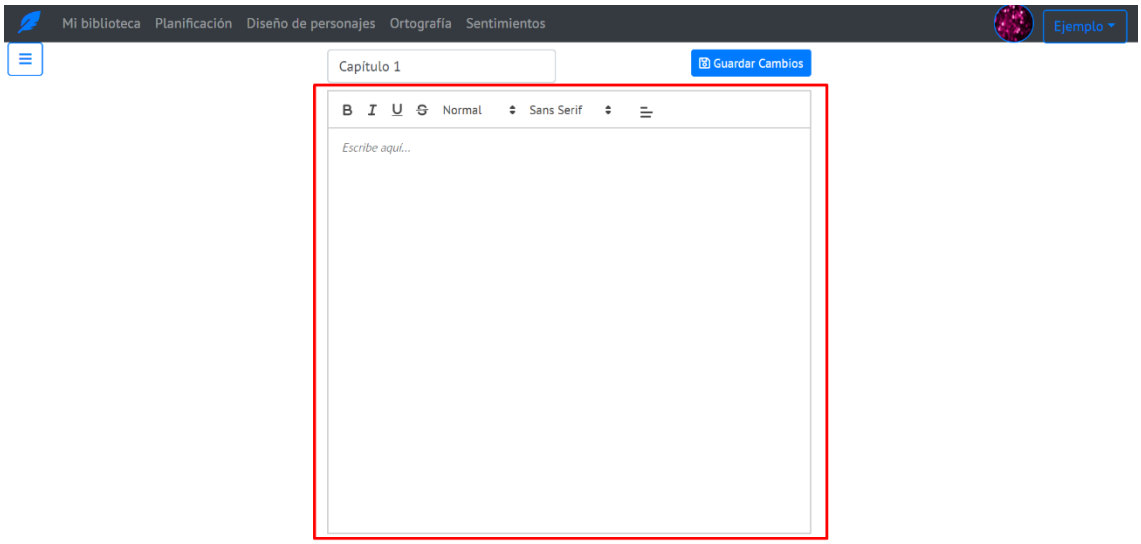


Figura 76 – Página de la escritura en línea

Una vez se tiene contenido escrito, pulsar sobre el botón “Guardar Cambios”.

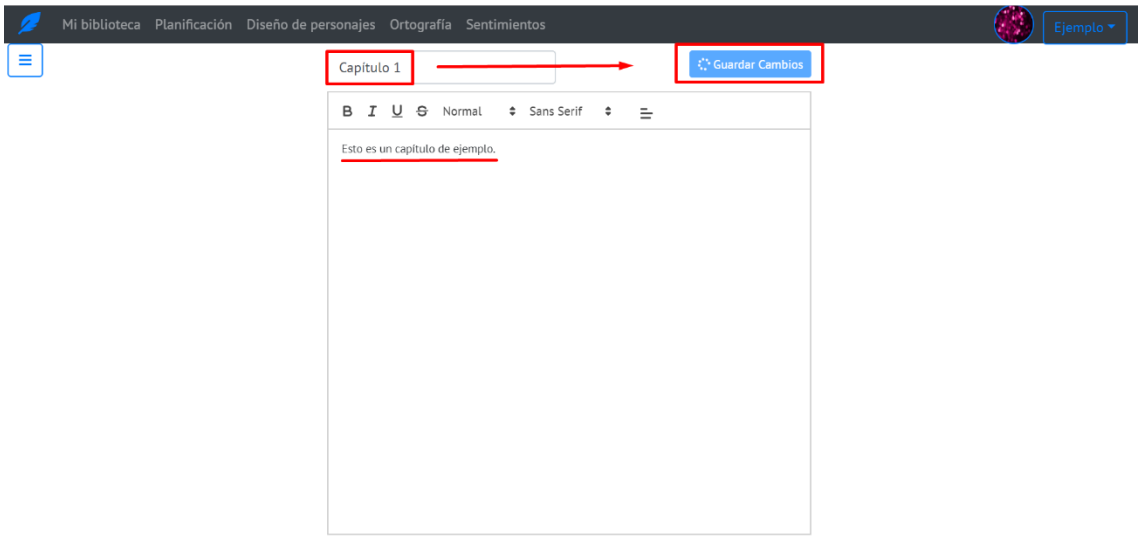


Figura 77 – Acciones en la escritura en línea

Tanto el título del capítulo como el contenido se guardan tras pulsar el botón.

Si se desea cambiar el formato, tamaño o apariencia del contenido escrito, pulsar sobre las opciones en la cabecera del editor.

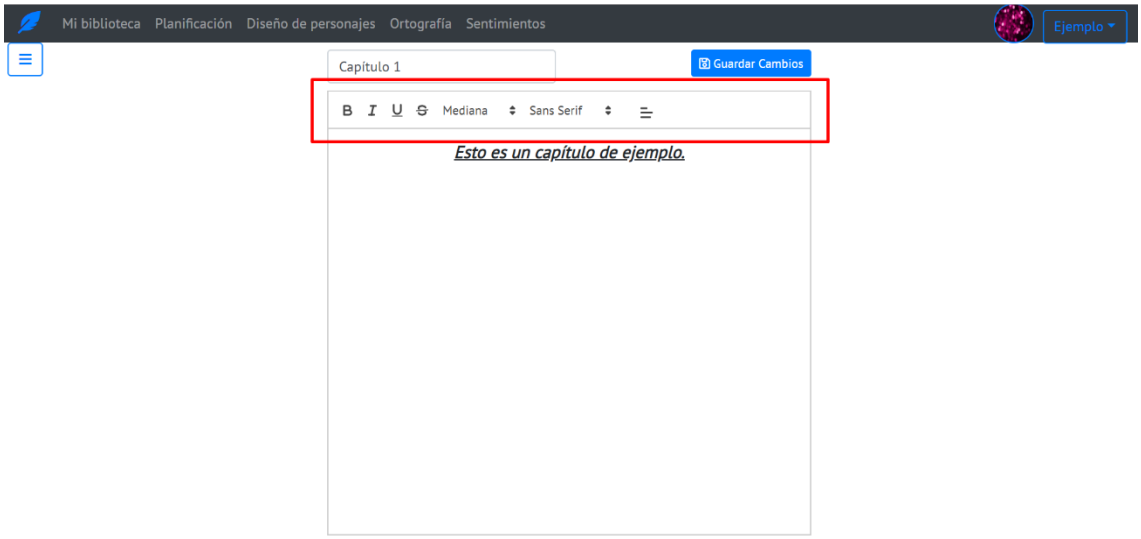


Figura 78 – Acciones de formato de texto en la escritura en línea

Cualquier cambio introducido, se guardará al pulsar sobre el botón de guardado.

Crear Guion Rápido

Desplegar herramientas pulsando el botón de menú desplegable situado arriba a la izquierda de la pantalla.

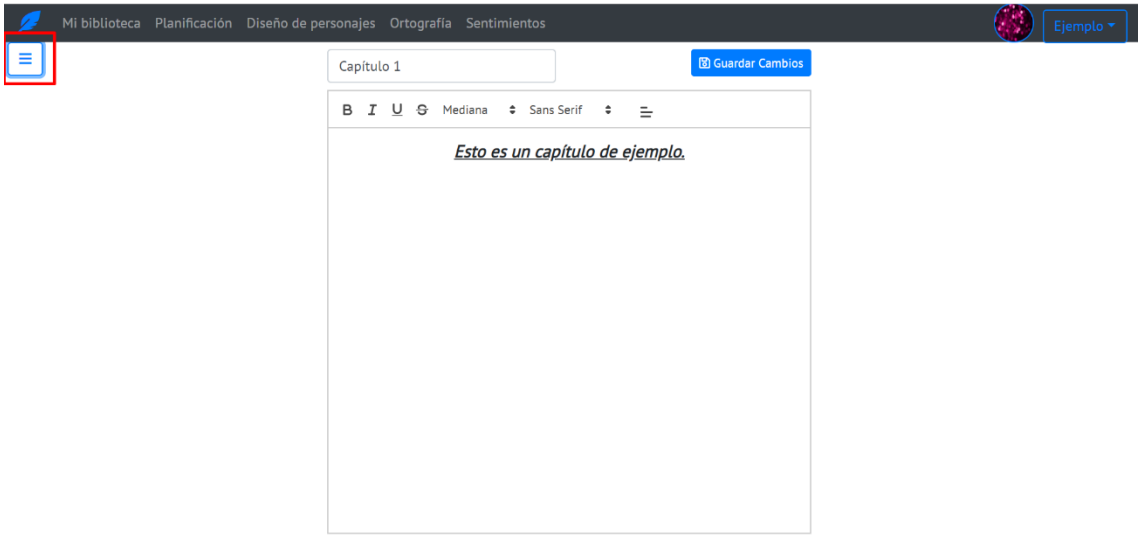


Figura 79 – Escritura en línea activando el guion rápido

Una vez se despliega, introducir el evento a planificar del capítulo y pulsar sobre el botón **+**, a lo que se verá añadido el evento a la lista.

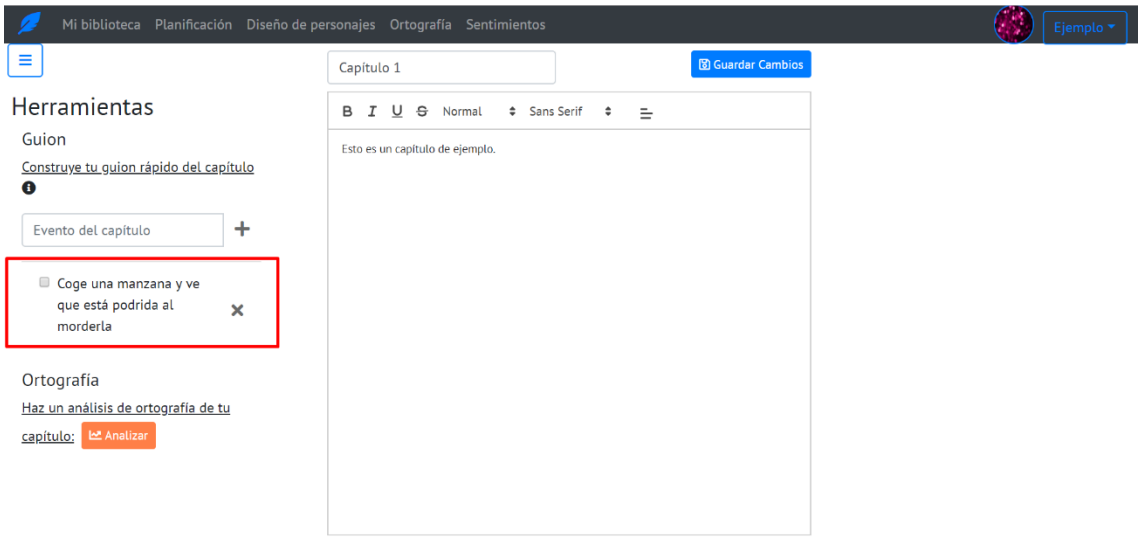


Figura 80 – Guion rápido con elementos

Para marcar el evento como completado pulsar sobre la caja de confirmación de la izquierda del evento.

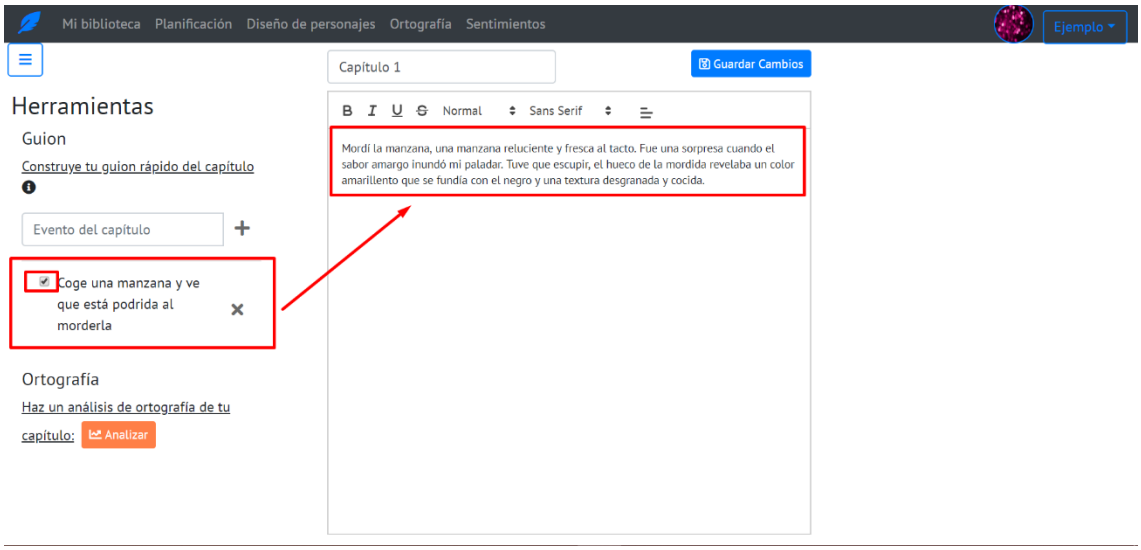


Figura 81 – Evento del guion rápido marcado

Para borrarlo basta con pulsar sobre el botón “X”

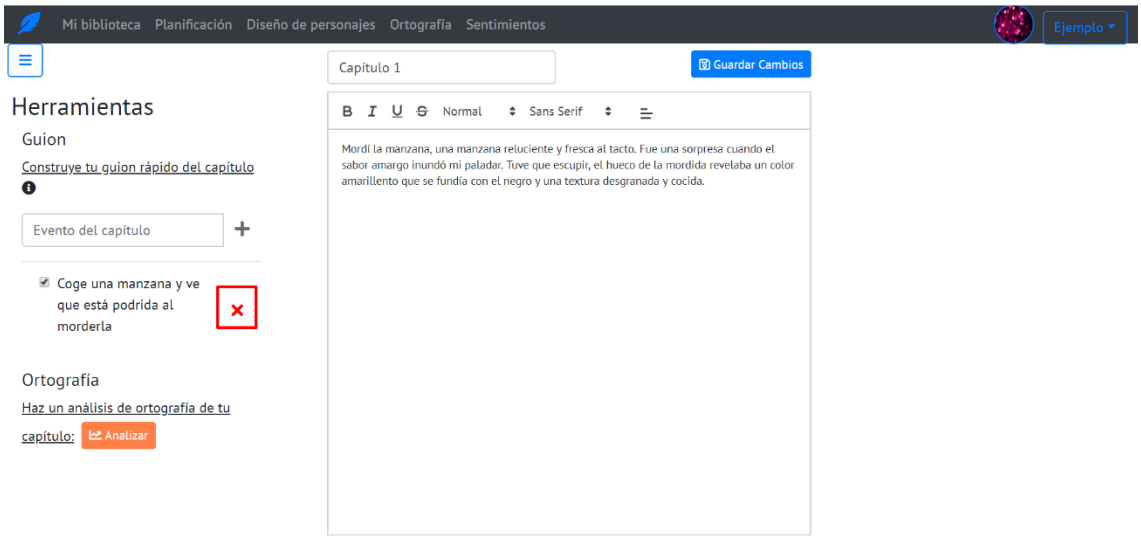


Figura 82 – Eliminar evento del guion rápido

Corrección de ortografía

Con el menú de herramientas desplegado, pulsar sobre el botón “Analizar”, bajo el apartado “Ortografía”.

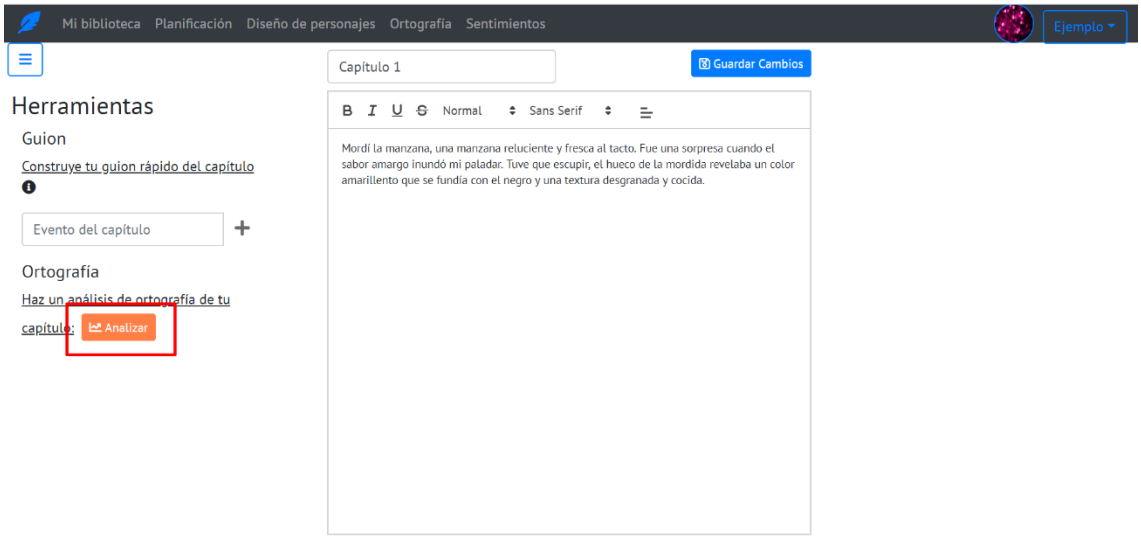


Figura 83 – Acción de análisis ortográfico en escritura en línea

Si detecta faltas, las mostrará en una lista, para ver las sugerencias y corregir la falta, pulsar sobre el elemento de la lista. Después pulsar “Corregir”.

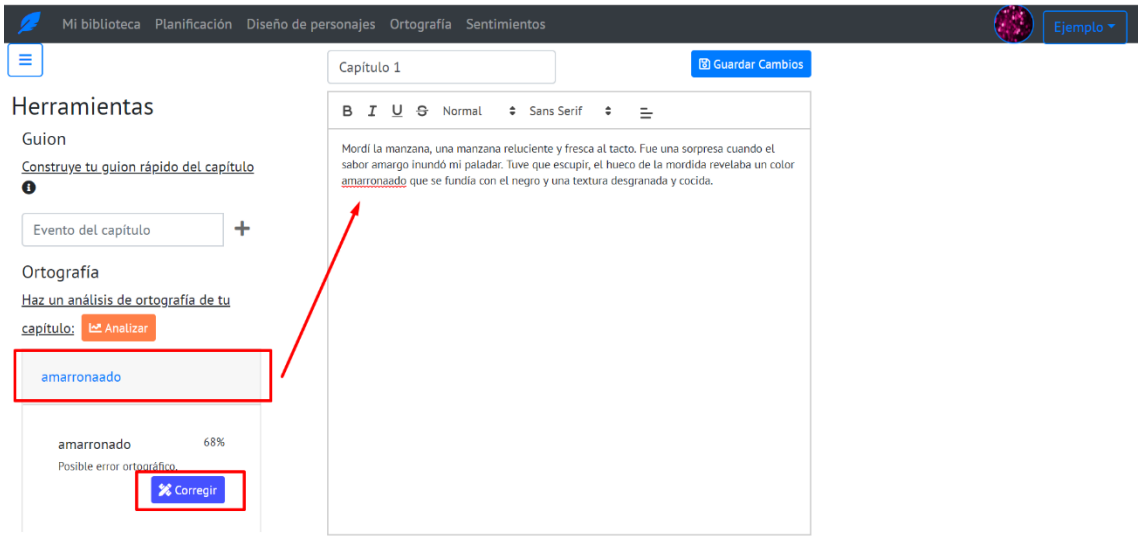


Figura 84 –Análisis ortográfico en escritura en línea

Tras pulsar:

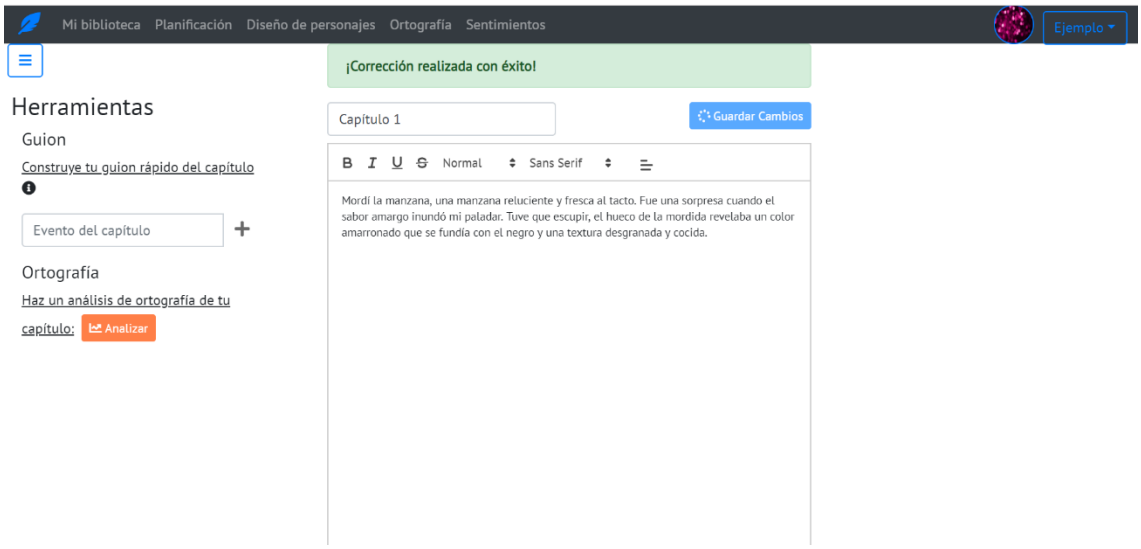


Figura 85 – Error corregido en la escritura en línea

Exportación de libro

Desde la pantalla de detalles del libro, pulsar sobre el botón “Exportar libro” de la carta de presentación.

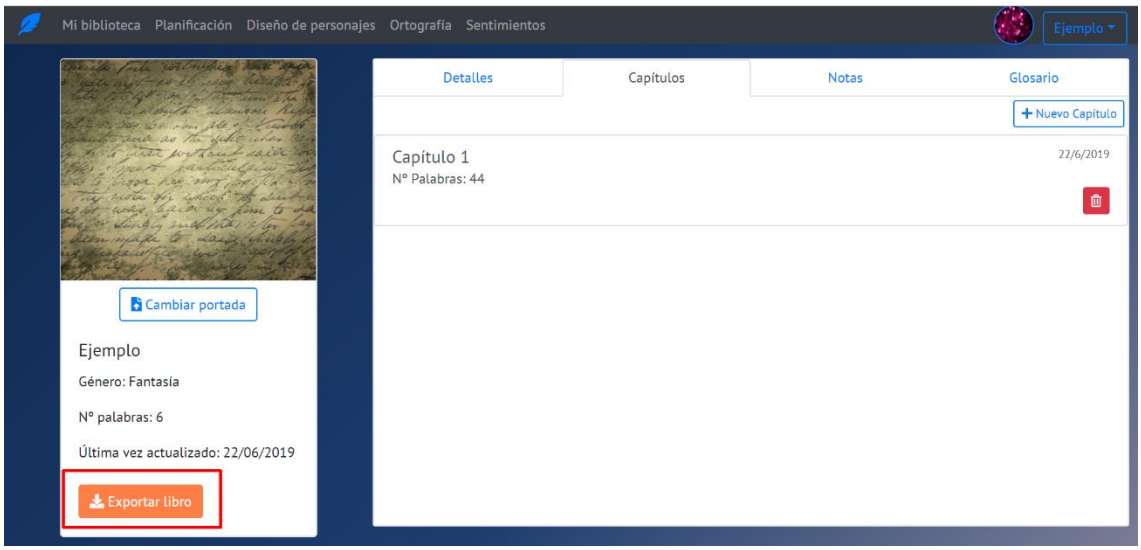


Figura 86 – Acción exportar libro

Se visualizará una descarga, pulsar sobre el documento y visualizar contenido.

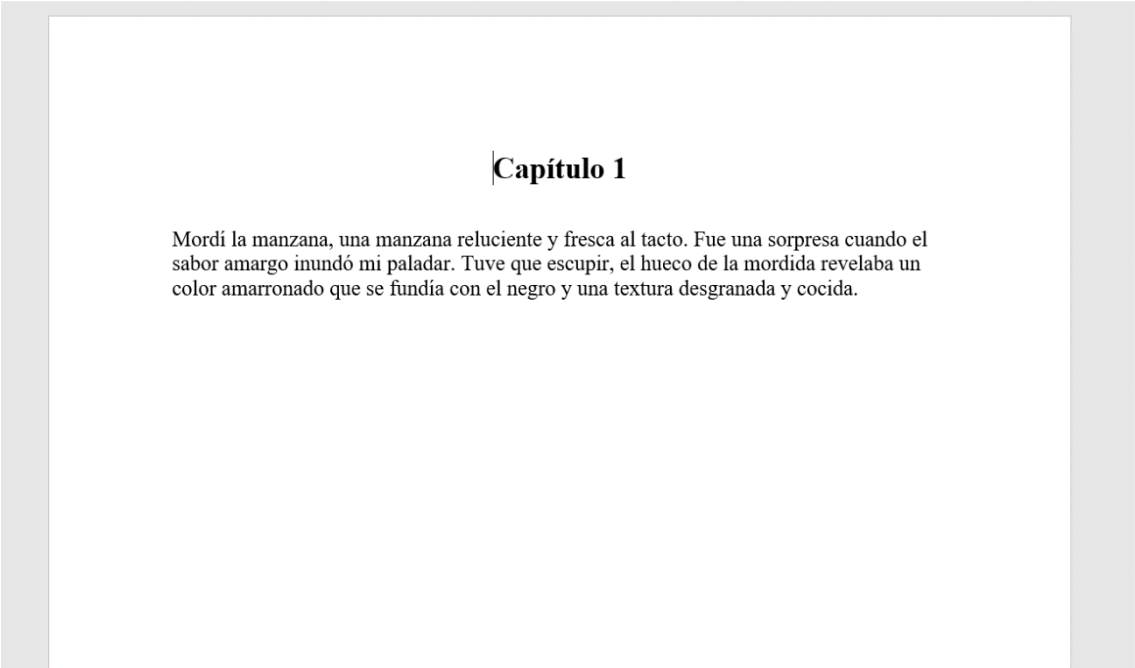


Figura 87 – Libro exportado a formato Word

Gráfica de análisis de ortografía

Pulsar sobre “Ortografía” en la barra de navegación.

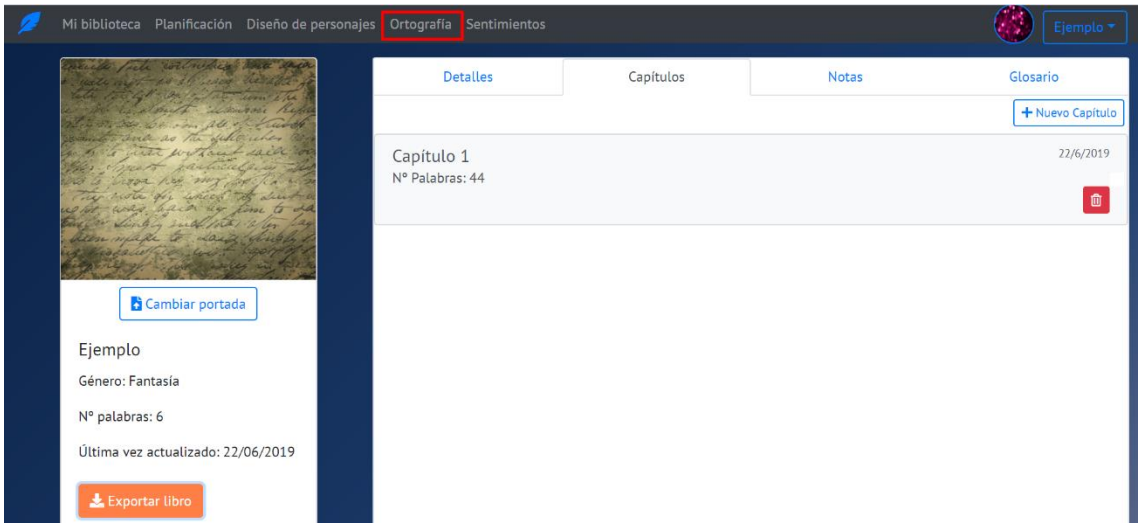


Figura 88 – Navegar a la página de ortografía

Se muestra una pantalla con un texto para escoger libro al que analizar. Selecciona el libro.

Figura 89 – Selección del libro a buscar faltas de ortografía

Tras ello, pulsar “Analizar”:

Figura 90 – Acción de análisis ortográfico

Detecta una falta, y a la derecha los capítulos que tienen la errata. Pulsar sobre el capítulo y proceder a corregir pulsando el botón “Corregir”

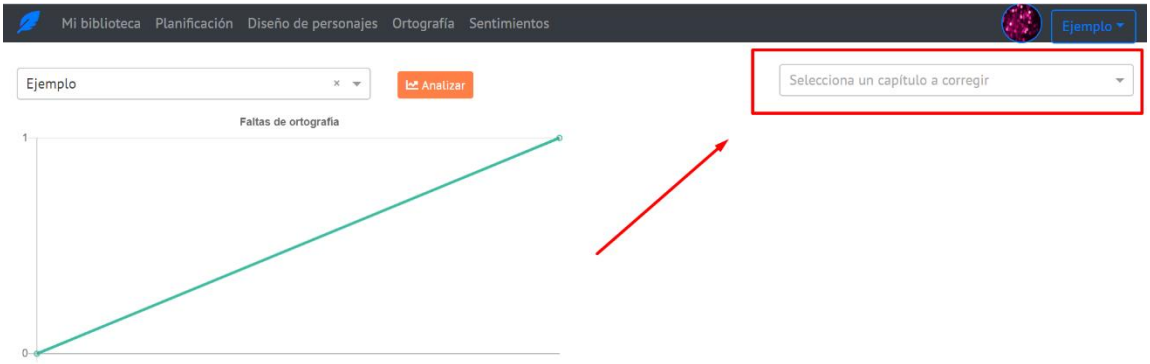


Figura 91 – Acción de selección del capítulo

Figura 92 – Acción corregir el error ortográfico

Figura 93 – Mensaje informativo de la corrección de la palabra

Análisis de Sentimientos

Seleccionar “Sentimientos” en la barra de navegación:

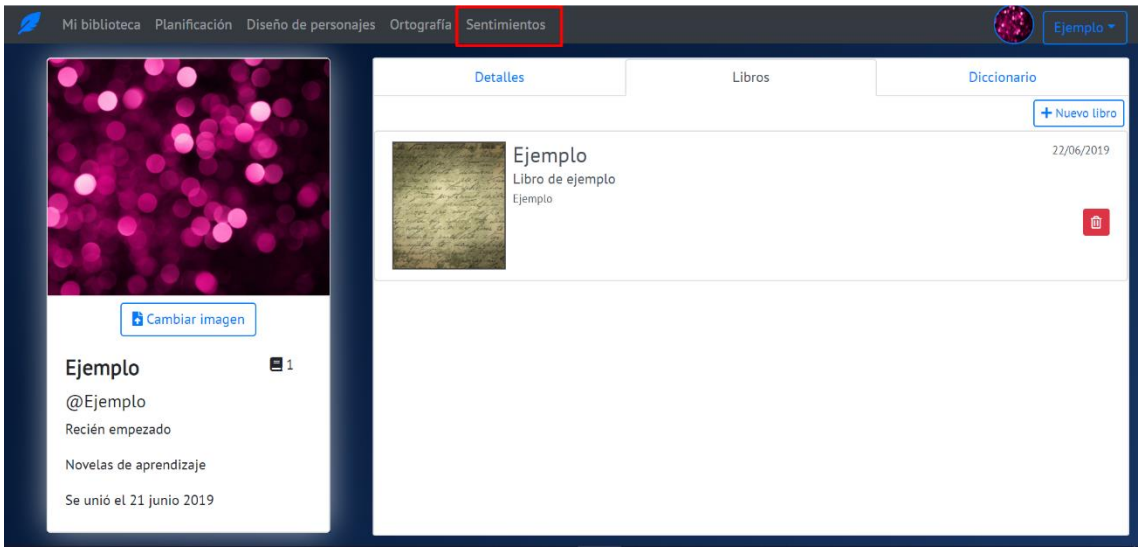


Figura 94 – Acción de navegar a la página de sentimientos

Aparece una pantalla con una gráfica de ejemplo, y un seleccionable con los libros sobre los que hacer análisis. Seleccionar el libro y pulsar “Analizar”:

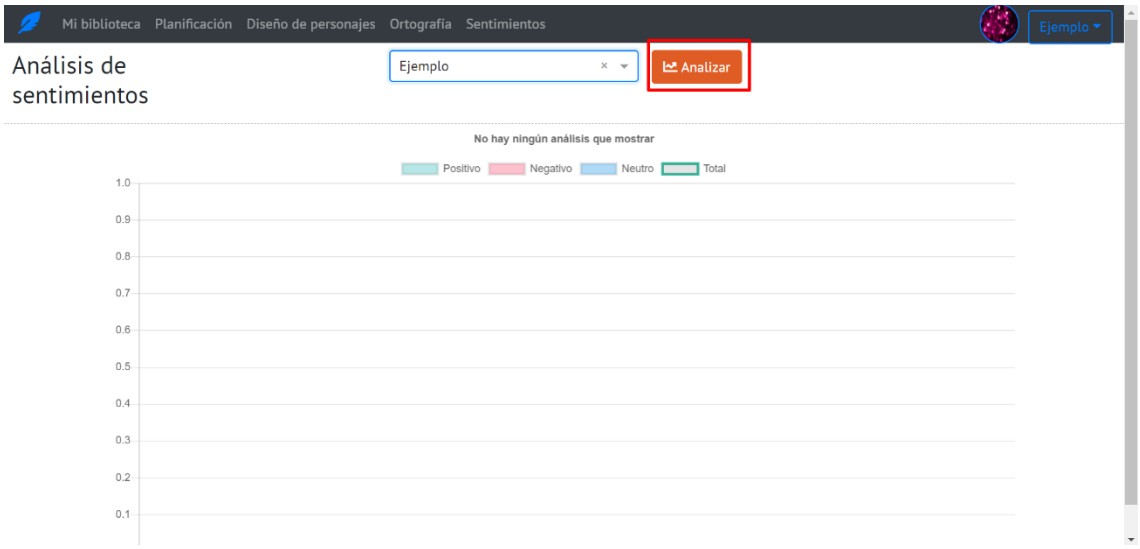


Figura 95 – Acción de análisis de sentimientos al libro seleccionado

Se visualiza el resultado:

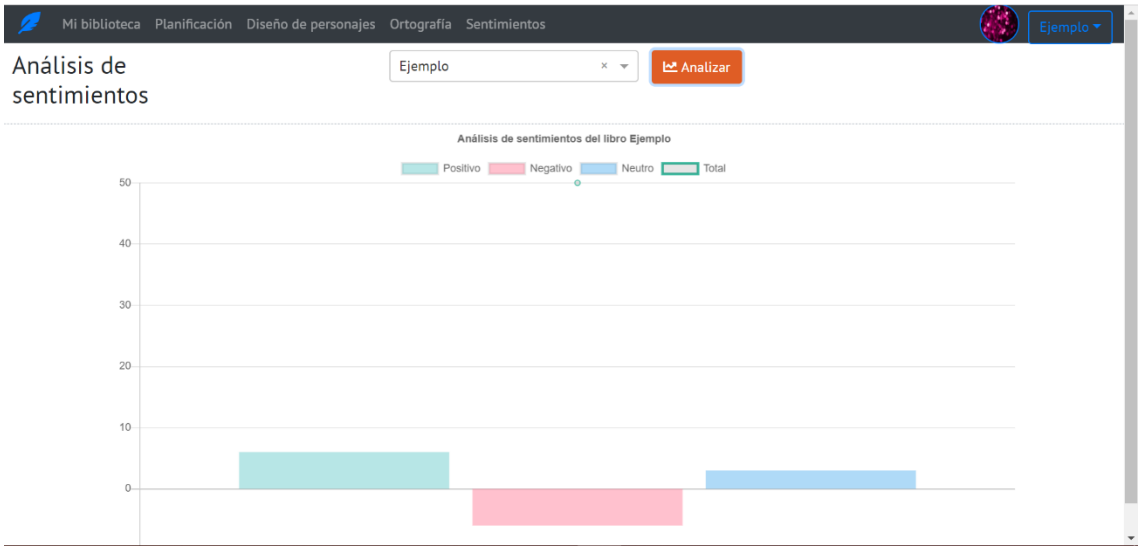


Figura 96 – Resultado del análisis de sentimientos

Diseño de personajes

Seleccionar “Diseño de personajes” en la barra de navegación

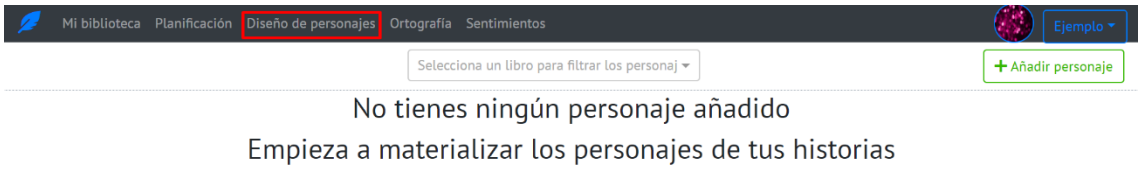


Figura 97 – Navegación a la galería de personajes

Se muestran los personajes creados por el autor. Para crear un personaje, pulsar “Añadir personaje”.

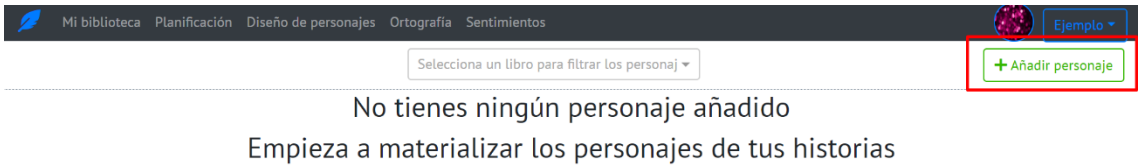


Figura 98 – Acción de crear un nuevo personaje

Se abre un formulario de creación. Para poder guardar el personaje se ha de introducir al menos el nombre y seleccionar el libro al que pertenece.

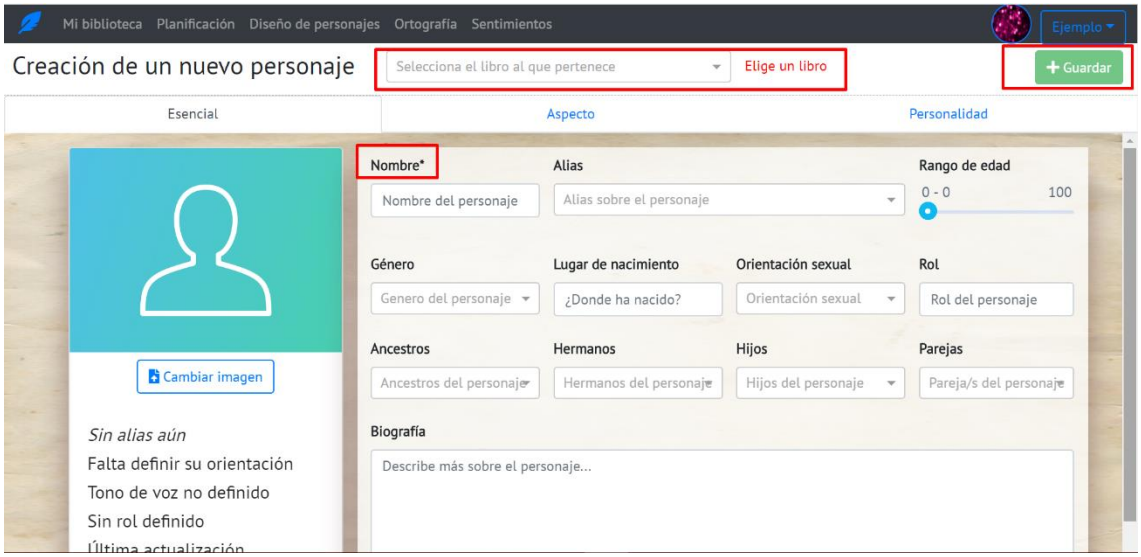


Figura 99 – Formulario de creación de personajes, esencial

Se presentan tres pantallas donde incluir información para el personaje, para acceder a cada una de ellas, seleccionar “Esencial”, “Aspecto” o “Personalidad”.

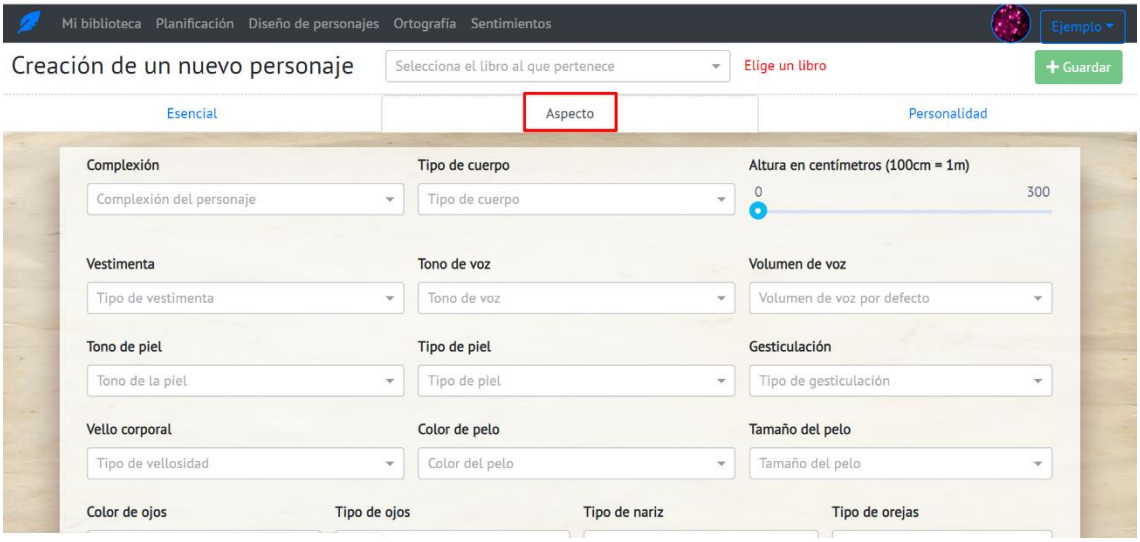


Figura 100 – Formulario de creación de personajes, aspecto

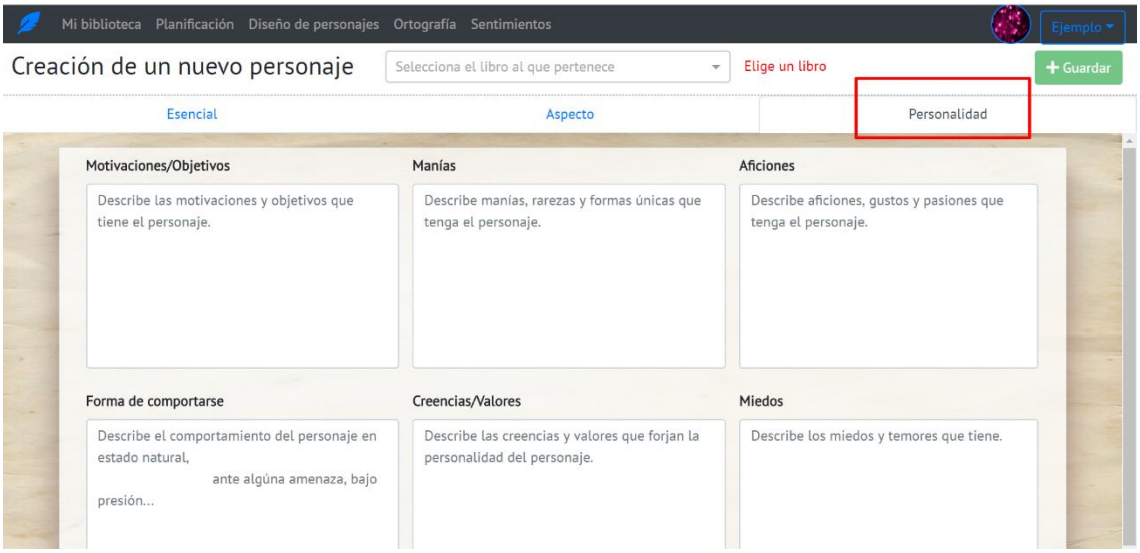


Figura 101 – Formulario de creación de personajes, personalidad

Pulsar “Guardar” al introducir la información:

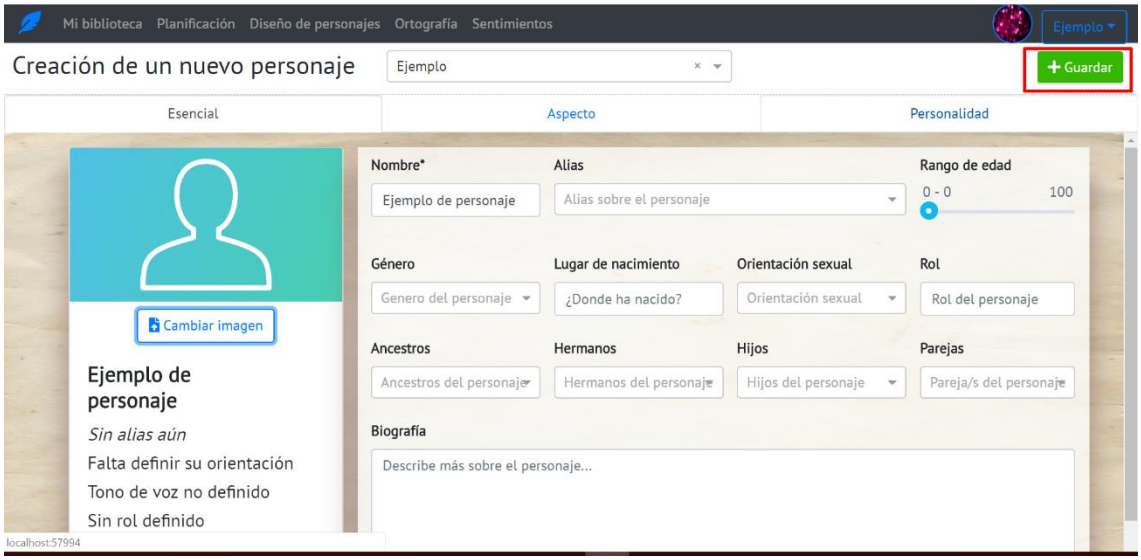


Figura 102 – Acción de guardar el personaje

Visualizar el personaje en la galería de personajes:

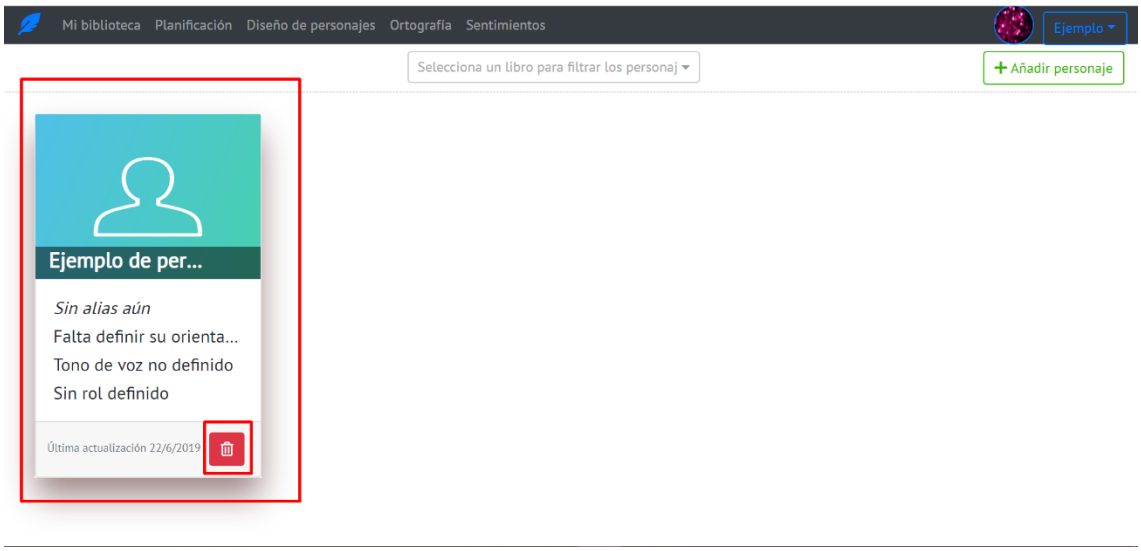


Figura 103 – Acciones de los personajes en la galería

Si se desea borrar, pulsar sobre el botón de la papelera y confirmar en la ventana de confirmación:

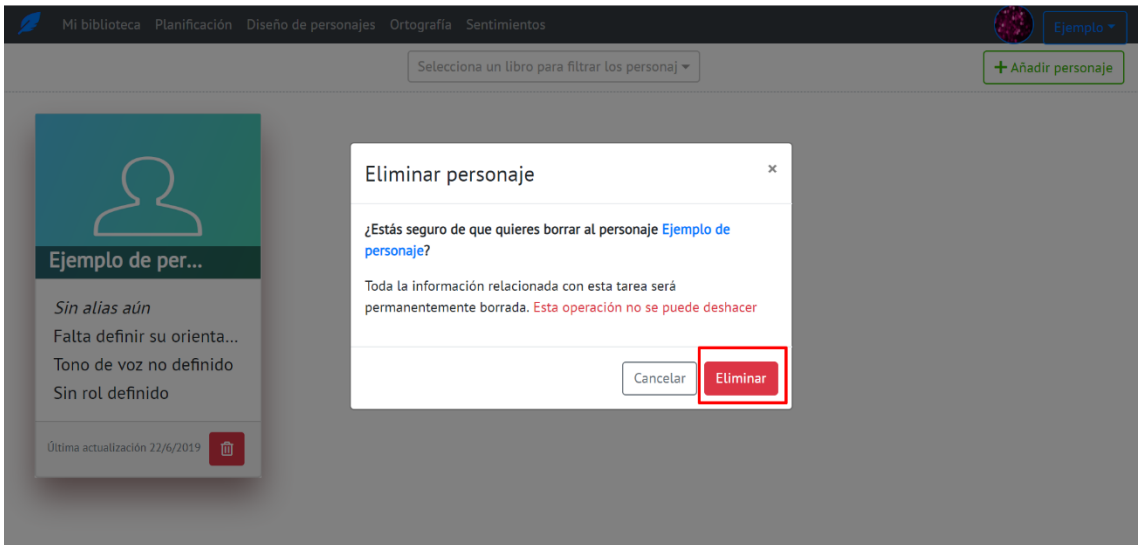


Figura 104 – Acción de eliminar un personaje